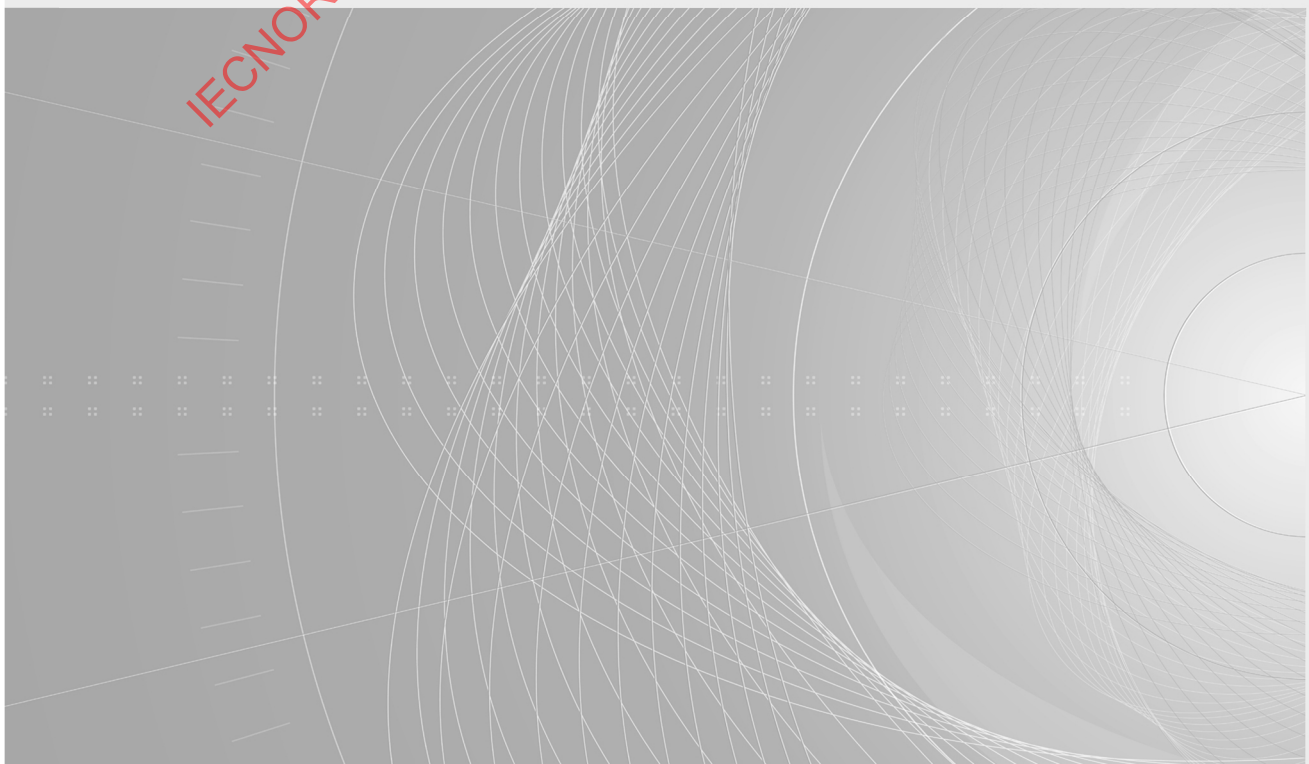
IEC 62455

Edition 2.0 2010-12

INTERNATIONAL STANDARD

colour inside

Internet protocol (IP) and transport stream (TS) based service access



IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 62455

Edition 2.0 2010-12

INTERNATIONAL STANDARD

colour inside

Internet protocol (IP) and transport stream (TS) based service access

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE XH

ICS 33.170; 35.100; 35.240.99

ISBN 978-2-88912-289-9

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INTERNET PROTOCOL (IP) AND TRANSPORT STREAM (TS) BASED SERVICE ACCESS

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62455 has been prepared by technical area 1: Terminals for audio, video and data services and content, of IEC technical committee 100: Audio, video and multimedia systems and equipment.

This second edition cancels and replaces the first edition, published in 2007, and constitutes a technical revision.

The main changes with respect to the previous edition are listed below.

- Recent developments in DVB and OMA standards caused some incompatibilities, which have been solved in the second edition.
- Technical errors have been corrected, missing details added.
- References have been updated to the newest available ones.
- In addition, a number of editorial corrections and readability improvements have been made, where the original text could have lead to misunderstanding due to unclear wording or the use of slightly different spellings for the same item.

The text of this standard is based on the following documents:

CDV	Report on voting
100/1551/CDV	100/1627/RVC

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTERNET PROTOCOL (IP) AND TRANSPORT STREAM (TS) BASED SERVICE ACCESS

1 Scope

This International Standard specifies the terminal for a service purchase and protection system for digital broadcasts, called the 18Crypt system. It is applicable in all countries and regions with suitably compliant broadcasting and multimedia distribution systems. Guidelines for compatible broadcast services are given in this standard. The service purchase and protection functions operate in a pure broadcast environment that may be combined with a bi-directional interactivity channel.

This standard is applicable to the following broadcast systems.

a) IP datacast over DVB-H systems

IP datacast over DVB-H is an end-to-end broadcast system for delivery of any type of digital content and services using IP-based mechanisms optimized for devices with limitations on computational resources and battery. An inherent part of the IP datacast system is that it comprises a unidirectional DVB broadcast path that may be combined with a bi-directional mobile/cellular interactivity path. IP datacast is thus a platform that can be used for enabling the convergence of services from broadcast/media and telecommunications domains (for example, mobile/cellular). This standard specifies service purchase and protection for IP datacast over DVB-H systems (see Table B.6 for an overview of references to one such system).

b) DVB T/C/S systems

DVB T/C/S systems are end-to-end broadcast systems for audio/video data that employ an MPEG2 transport stream and use terrestrial, cable or satellite broadcast networks. This standard specifies a system for the protection of these broadcasts in a pure broadcast environment. In addition, this standard specifies how purchasing, key management and registration may be carried out over an optional interactivity channel. The protection technologies offered by this standard are designed to operate within an existing DVB SimulCrypt environment (see Table B.6 for an overview of references).

c) MPEG2 TS-based IP systems

MPEG2 TS-based IP systems employ bi-directional IP networks for the (broadcast) delivery of MPEG2 transport streams. This standard specifies a system for the purchase and protection of services and content delivered via these networks. This standard is applicable to, for example, DVB-IPI systems (see Table B.6 for an overview of references).

d) Non-MPEG2 TS-based IP systems

Non-MPEG2 TS-based IP systems employ bi-directional IP networks for the (broadcast) delivery of audio/video or other data using IP protocols instead of an MPEG2 transport stream. This standard specifies a system for the purchase and protection of services and content delivered via these networks (see Table B.6 for an overview of references).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 8859-1:1998, *Information technnology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO/IEC 13818-1:2007, *Information technology – Generic coding of moving pictures and associated audio information: Systems*

ISO/IEC 14496-12:2008, *Information technology – Coding of audio-visual objects – Part 12: ISO base media file format*

ISO/IEC 15938-5:2003, *Information technology – Multimedia content description interface – Part 5: Multimedia description schemes*

ISO 639-1:2002, *Codes for the representation of names of languages – Part 1: Alpha-2 code*

ISO 639-2:1998, *Codes for the representation of names of languages – Part 2: Alpha-3 code*

ISO 3166 (all parts), *Codes for the representation of names of countries and their subdivisions*

ISO 4217, *Codes for the representation of currencies and funds*

ISO 8601:2004, *Data elements and interchange formats – Information interchange – Representation of dates and times*

ETSI EN 102 034, *Digital Video Broadcasting (DVB) – Transport of MPEG-2-based DVB services over I- based networks*

ETSI EN 300 468, *Digital Video Broadcasting (DVB) – Specification for Service Information (SI) in DVB systems*

ETSI EN 301 192, *Digital Video Broadcasting (DVB) – DVB specification for data broadcasting*

ETSI EN 302 304, *Digital Video Broadcasting (DVB) – Transmission system for handheld terminals (DVB-H)*

ETSI TS 102 539, *Digital Video Broadcasting (DVB) – Carriage of broadband content guide (BCG) information over internet protocol (IP)*

ETSI ETR 162, *http://www.dvb.org/products_registration/dvb_identifiers/*(this website replaces ETR 162)

ETSI ETR 289, *Digital Video Broadcasting (DVB) – Support for use of scrambling and conditional sccess (CA) within digital broadcasting systems*

ETSI TS 102 471, *Digital Video Broadcasting (DVB) – IP datacast over DVB-H: Electronic service guide (ESG)*

ETSI TS 102 472, *Digital Video Broadcasting (DVB) – IP datacast over DVB-H: Content delivery protocols*

ETSI TS 102 822-3-1, *Broadcast and on-line services: Search, select, and rightful use of content on personal storage systems (TV-anytime) – Part 3: Metadata – Sub-part 1: Phase 1 – Metadata schemas*

ETSI TS 103 197, *Digital Video Broadcasting (DVB) – SimulCrypt Head-end implementation of DVB SimulCrypt*, v1.4.1

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1.1
18Crypt
service purchase and protection system specified by this standard

3.1.2
broadband content guide
service guide in DVB-IPI based IP systems; see ETSI EN 102 034

3.1.3
broadcast channel
broadcast network

NOTE See also other definitions using "broadcast".

3.1.4
broadcast device
device that is capable of receiving protected broadcasts but which cannot access an interactivity channel

NOTE All messages sent to such a device are delivered via a broadcast channel and all communications from the device to rights issuers is done out of band.

3.1.5
broadcast domain
group of devices for which rights objects granting the same rights to the whole group can be issued via a broadcast channel

3.1.6
broadcast network
digital transmission system supporting only unidirectional communication from the broadcaster to multiple devices

3.1.7
broadcast rights object
content, programme or service rights object delivered over the broadcast network

3.1.8
broadcast service
service carried over a broadcast network

3.1.9
broadcast
unidirectional distribution to all receivers

3.1.10
conditional access
encryption/decryption management method (security system) where the broadcaster controls the subscriber's *access* to digital services

3.1.11
content
any form of digital media that can be acquired and presented by a device

3.1.12
content on demand
delivery of content to a single device as a result of a specific request from a user

3.1.13
content protection
protection of content such that it can only be presented by authorized devices

3.1.14
content rights object
rights object containing a content key and pertaining to content protection

3.1.15
customer operation centre
entity responsible for billing the customer

NOTE It is assumed that the customer will have a contractual relationship with this organization.

3.1.16
device
terminal that is capable of receiving broadcast transmission over broadcast network

NOTE The terminal may be capable of interacting with a service provision subsystem and a commerce subsystem through a cellular network. The terminal may include optional embedded secure hardware, such as a UICC.

3.1.17
digital rights management
set of methods which ensures that content, during its entire lifetime, can only be used in specified ways and only when the relevant conditions (for example, access conditions) have been met

NOTE These "specified ways" and "relevant conditions" are expressed in rights objects.

3.1.18
DRM time
secure, non user-changeable time source

NOTE The DRM time is measured in the UTC time scale.

3.1.19
DVB network
network for transmitting a collection of MPEG-2 transport streams, each carrying a multiplex, and transmitted on a single delivery system

NOTE DVB network is identified by network_id.

3.1.20
electronic service guide
service guide in IPDC over DVB-H systems

3.1.21
interactive device
device that is capable of directly connecting to a rights issuer using an appropriate protocol over an interactivity channel over an appropriate transport/network layer interface, for example, HTTP over TCP/IP

3.1.22
interactive domain
OMA DRM 2.0 domain, as specified in OMA-ERP-DRM-V2_0

NOTE This is a group of devices for which rights objects can be issued via an interactivity channel granting rights to the whole group.

3.1.23
interactivity channel rights object
content, programme or service rights object delivered over an interactivity channel

3.1.24
interactivity channel
bi-directional channel established between the broadcast service network and the device for reliable exchange of messages

3.1.25
interactivity network
network supporting bi-directional communication and the reliable delivery of messages between the device and the rights issuer

3.1.26
interactivity network cell
cell forming part of an interactivity network

3.1.27
IP flow
stream of IP datagrams each sharing the same IP source and destination address

3.1.28
IPDCKMSId
identifier assigned for a particular IPDC key management system by DVB (also known as CA_system_id in ETSI ETR 162)

3.1.29
ISMACryp
end-to-end content encryption system for media carried over RTP streams and ISO-based media files

3.1.30
key management system
end-to-end system to authorize users and provide them the necessary means to access protected content

3.1.31
key stream message
message broadcast alongside a protected service, carrying key material to decrypt and optionally authenticate the service

3.1.32
media access unit
smallest data entity to which timing information can be attributed

NOTE In the case of audio, an access unit is an audio frame, and in the case of video, a picture.

3.1.33
media flow
elementary stream or IP flow carrying audio or video data

3.1.34
mixed-mode device
combination of a broadcast device and an interactive device

3.1.35
mobile TV application
main device application responsible for accessing the mobile TV services

NOTE This is sometimes referred to as the player application.

3.1.36
nonce
randomly chosen value, different from previous choices, inserted in a message to protect against replay attacks

3.1.37
off line
without using a direct link (such as the interactivity channel) between network entities and the device (also called "out of band")

3.1.38
OMA BCAST
working group in OMA developing a fully standardized system allowing operators to control access and usage of broadcast content on mobile devices

3.1.39
out of band
without using a direct link (such as the interactivity channel) between network entities and the device (also called "off line")

NOTE The name OOB is used in different ways in different contexts. OOB is used to signal out of band notification of data to the RI, as, for example, in registration over means other than the interactivity channel. In DVB-T/C/S systems, the term OOB or "out of band" is well known to signal a part of the spectrum other than the in band (a.k.a. IB). In the latter context, the OOB may carry data in such environments to save bandwidth for repeating messages in the IB. In the T/C/S context, DVS-167/178 or DOCSIS are well-known "OOB" systems. To make matters more complex, DOCSIS may be used as OOB when multiplexing in tables in the downstream and as interactivity channel when carrying IP data packets in Ethernet frames.

3.1.40
parameter set
either a sequence parameter set or a picture parameter set

NOTE This term is used to refer to both types of parameter sets.

3.1.41
parameter set elementary stream
elementary stream containing access units made up of NAL units for coded picture data

3.1.42
programme rights object
rights object containing a programme key or keys and pertaining to service protection

3.1.43
programme
logical portion of a service with a distinct start and end

3.1.44
rights issuer context
context consisting of information that was negotiated with a given rights issuer, during the 4-pass registration protocol such as the rights issuer ID, rights issuer certificate chain, version, algorithms used and other information

NOTE A rights issuer context is necessary for a device to participate successfully in all the protocols of the ROAP suite, except the registration protocol.

3.1.45
rights issuer service
IP datacast channel used to carry broadcast rights objects, registration data and other messages from a rights issuer over a broadcast channel

3.1.46
rights issuer
entity that registers devices and provides rights objects allowing devices to receive protected services

3.1.47
rights object
collection of permissions, keys and other attributes that are linked to items of content or services

3.1.48
roaming
end-user accessing IPDC services through another ('foreign') service provider than the 'home' service provider

NOTE Roaming is usually based on roaming agreements between service providers.

3.1.49
service
one or more IP flows intended to be presented together

NOTE Examples include, but are not limited to, audio and video services.

3.1.50
service guide
data source describing the services and content available via a network and how they can be received

3.1.51
service operation centre
service provider

NOTE See also other definitions with 'service'.

3.1.52
service protection
protection of a service such that only authorized devices are able to receive and decode it

3.1.53
service provider
provider of a broadcast service – the entity that broadcasts the service, and provides key and schedule information to rights issuers

3.1.54
service rights object
rights object containing one or multiple service keys and pertaining to service protection

3.1.55
service roaming
roaming during which the user has access to the same services as through the home service provider

3.1.56
simulCrypt
system that provides the ability for multiple KMSs to control access at the same time to a single content stream

3.1.57
super distributable content
protected (encrypted) content that can be freely distributed and for which the recipients can obtain ROs from an RI for the consumption of that content, because that protected (encrypted) content has a link to the RI in it as well as enough information such that the RI can determine the decryption key

3.1.58
terminal
consumer device that can receive, descramble, and decode services

3.1.59
token rights object
special interactivity channel rights object containing tokens for pre-/post-paid consumption-based charging of interactive devices, or a special broadcast rights object containing tokens for pre-/post-paid consumption-based charging of broadcast devices

3.1.60
unconnected device
device, as defined in OMA-AD-DRM-V2_0, which cannot connect directly to a rights issuer via an interactivity channel for the acquisition of rights objects, but can do so via an intermediary device

3.1.61
user
person using the device to receive protected services

3.1.62
user roaming
roaming during which the user has access to the services of the foreign service provider

3.1.63
video elementary stream
elementary stream containing samples made up of only sequence and picture parameter set NAL units synchronized with the video elementary stream

3.2 Symbols

For the purposes of this document, the following symbols apply.

E{K}(M)	Encryption of message ‘M’ using key ‘K’
D{K}(M)	Decryption of message ‘M’ using key ‘K’
P	Public key
Q	Private key
RIQ	Rights issuer private key
RIP	Rights issuer public key
DQ	Device private key
DP	Device public key
A{K}(M)	Authentication of message ‘M’ with key K
V{K}(M)	Verification of message ‘M’ with key K
A==B	Yields true if and only if A equals B
A&&B	Yields true if and only if both A and B are true

A & B	Bit-wise AND of A and B
A ⊕ B	Bit-wise exclusive OR of A and B
A | B	Bit-wise OR of A and B
A || B	Concatenation of A and B
A + B	Arithmetic addition in case A and B are numerical arguments; concatenation of A and B if A and B are strings
A << B	Bit-wise shift left operation of A by B bits, filling with 0s
A >> B	Bit-wise shift right operation of A by B bits, filling with 0s
LSBm(X)	The bit string consisting of the *m* least significant bits of the bit string *X*
MSBm(X)	The bit string consisting of the *m* most significant bits of the bit string *X*

3.3 Abbreviations

For the purposes of this document, the following abbreviations apply.

ACG	Access Criteria Generator
ADSL	Asymmetric Digital Subscriber Line
AES	Advanced Encryption Standard
AID	Application Identifier
APDU	Application Protocol Data Units
API	Application Program Interface
ARC	Action Request Code
ASCII	American Standard Code for Information Interchange
ATM	Asynchronous Transfer Mode
AU	Access Unit
AVC	Advanced Video Codec
AVP	Attribute-Value Pairs
BAK	BCRO Authentication Key
BCD	Binary Coded Decimal
BCG	Broadband Content Guide
BCI	Binary Content ID
BCRO	Broadcast Rights Object
BDK	Broadcast Domain Key
BOC	Broadcast Operation Centre
bslbf	bit string left bit first
CA	Conditional Access
CAS	Conditional Access System
CAT	Conditional Access Table
CBC	Cipher Block Chaining
CCI	Copy Control Information
CDC	Connected Device Configuration
CDP	Content Delivery Protocol
CEK	Content Encryption Key
CGF	Customer Group Filter
CLDC	Connected Limited Device Configuration
CID	Content ID
CIEK	Content Item Encryption Key
COC	Customer Operation Centre
CoD	Content on Demand
CPCM	Content Protection and Copy Management

CRC	Cyclic Redundancy Check
CRL	Certificate Revocation List
CSA	Common Scrambling Algorithm
CTR	Counter
DCF	DRM Content Format
DIST Mgmt	Service Distribution Management
DOCSIS	Data Over Cable Service Interface Specification
DRD	Device Registration Data
DRM	Digital Rights Management
DVB	Digital Video Broadcasting
DVB-C	DVB-Cable
DVB-H	DVB-Handheld
DVB-IPI	DVB-Internet Protocol Infrastructure
DVB-S	DVB-Satellite
DVB-T	DVB-Terrestrial
ECB	Electronic Code Book
ECM	Entitlement Control Message
ECMG	ECM Generator
EIS	Event Information System
EIT	Event Information Table
EMM	Entitlement Management Message
EMMG	EMM Generator
EPG	Electronic Programme Guide
ESG	Electronic Service Guide
ESP	Encapsulating Security Payload
GCF	Generic Connection Framework
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HMAC	Hashed Message Authentication Code
HO	Home Operator
HTTP	Hyper Text Transfer Protocol
HTTPS	Secure – Hyper Text Transfer Protocol
IC	Interactivity Channel
ICRO	Interactivity Channel Rights Object
ID	Identifier
IEK	Inferred Encryption Key
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
IPDC	IP DataCast
IPPV	Impulse Pay Per View
IPsec	IP Security
IPTV	Internet Protocol Television
IRD	Inform Registered Device (protocol)
IrDA	Infrared Data Association
ISMA	Internet Streaming Media Alliance
ISO	International Standards Organization
IV	Initial Vector
KML	Key Management Layer
KMM	Key Management Message

KMS	Key Management System
KSL	Key Stream Layer
KSM	Key Stream Message
LBDF	Long-form Broadcast Domain Filter (a.k.a. longform_domain_id)
lsb	least significant bit
MAC	Message Authentication Code
MF	Master File
MII	Major Industry Identifier
MIME	Multipurpose Internet Mail Extensions
MJD	Modified Julian Date
mjdutc	Modified Julian Date UTC
MK	Master Keys
MKI	Master Key Indicator
MPEG	Moving Pictures Experts Group
MPTS	Multi Program Transport Stream
msb	most significant bit
MSISDN	Mobile Station Integrated Services Digital Network
MTU	Maximum Transmission Unit
NAL	Network Abstraction Layer
NDD	Notification of Detailed Data
NIST	National Institute of Standards and Technology
NIT	Network Information Table
NSD	Notification of Short Data
OBEX	Object Exchange Protocol
OCSP	Online Certificate Status Protocol
OMA	Open Mobile Alliance
OOB	Out Of Band
OTA	Over The Air (i.e. transfer over a wireless connection)
PAK	Programme Authentication Key
PAS	Programme Authentication Seed
PAT	Programme Association Table
PCT	Purchase Channel Table
PDCF	Packetised DRM Content Format
PDR	Push Device Registration
PEAK	Programme Encryption/Authentication Key/Seed
	NOTE Concatenation of PEK and PAS.
PEK	Programme Encryption Key
PES	Packetised Elementary Stream
PID	Packet Identifier
PIT	Purchase Item Table
PKC	Public Key Certificate
PKCS	Public-Key Cryptography Standards
PKC-ID	PKC Identifier: the hash of the Public Key Certificate
PKI	Public Key Infrastructure
PMT	Programme Map Table
PPV	Pay Per View
PS	Program Stream
PSI	Programme Specific Information
PSS	Probabilistic Signature Scheme

RI	Rights Issuer
RIAK	Right Issuer Authentication Key
RID	Registered application provider Identifier
RIS	Rights Issuer Services
RML	Rights Management Layer
RO	Rights Object
ROAP	Rights Object Acquisition Protocol
ROC	Roll-Over Counter
ROT	Root Of Trust
rpchof	remainder polynomial coefficients, highest order first
RSA	Rivest-Shamir-Adleman public key algorithm
RSASSA-PSS	RSA Signature Scheme with Appendix - Probabilistic Signature Scheme
RTP	Real-time Transport Protocol
SA	Security Association
SAC	Secure Authenticated Channel
SAK	Service Authentication Key
SAS	Service Authentication Seed
SBDF	Short-form Broadcast Domain Filter (a.k.a. shortform_domain_id)
SBT	Service Bundle Table
SCS	SimulCrypt Synchronizer
SDP	Session Description Protocol
SDT	Service Description Table
SEAK	Service Encryption / Authentication Key / Seed
	NOTE Concatenation of SEK and SAS.
SEK	Service Encryption Key
SG	Service Guide
SGK	Subscriber Group Key
SHA-1	Secure Hash Algorithm
SI	Service Information
SIM	Subscriber Identity Module
SK	Session Key
SMS	Short Message Service
SOC	Service Operation Centre
SPI	Security Parameters Index
SPP	Service Purchase and Protection
SRTP	Secure Real-Time Transport Protocol
SUB Mgmt	Service Subscription Management
TAA	Trust Authority Algorithm
TAK	Traffic Authentication Key
TAS	Traffic Authentication Seed
TDK	Token Delivery Key
TDRMAK	Token Delivery Response Message Authentication Key
TEK	Traffic Encryption Key
TS	Transport Stream
UDF	Unique Device Filter
UDK	Unique Device Key
UDN	Unique Device Number
UDP	User Datagram Protocol
UGK	Unique Group Key

UICC	Universal Integrated Circuit Card
uimsbf	unsigned integer, most significant bit first
UMTS	Universal Mobile Telecommunications System
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
USIM	Universal Subscriber Identity Module
UTC	Universal Time Coordinated
UTF	Unicode Transformation Format
UTF-8	8-bit Unicode Transformation Format
VO	Visited Operator
VM	Virtual Machine
WAP	Wireless Application Protocol
WIM	WAP Identify Module
XML	eXtensible Markup Language
ZMB	Zero Message Broadcast

3.4 Identifiers assigned by external entities

18CRYPT_CAS_ID: This is the CA_system_id assigned to 18Crypt by the DVB project. The value can be found at

ETSI ETR 162: http://www.dvb.org/products_registration/dvb_identifiers/

18CRYPT_PRIVATE_DATA_ID: This is the private data identifier ID assigned to 18Crypt by the DVB project. The value can be found at

ETSI ETR 162: http://www.dvb.org/products_registration/dvb_identifiers/

4 General

4.1 Overview

This clause provides a general introduction to this standard. It covers

- the service purchase and protection system,
- the four-layer model for service protection,
- the end-to-end system and use of broadcast and interactivity channels,
- the types of systems and devices supported by this standard,
- difference between service and content protection.

The specification subsequent to the present clause is structured as follows.

- Clause 5 provides general specifications and concepts for the service purchase and protection system.
- The four-layer model is specified in 5.4. The four layers are specified in Clauses 6, 7, 8 and 9 respectively.
- Clause 10 specifies signalling and service and content discovery (service guide – SG).
- Clause 11 specifies the RI services, i.e. the system with which messages meant for individual devices or groups of devices are broadcast.

- Service purchase is specified in Clause 12.
- The subsequent clauses provide normative prescriptions specific for the individual supported systems, which are:
  - IPDC over DVB-H systems, Clause 13;
  - DVB T/C/S systems, Clause 14;
  - MPEG2 TS-based IP systems, Clause 15;
  - Non-MPEG2 TS-based IP systems, Clause 16.

Annex A normatively specifies several required tools and technologies in detail.

Annex B may provide the reader with more insight into the system.

4.2 General description of the system and elements

4.2.1 General

This standard specifies a service purchase and protection system for services transmitted over a broadcast infrastructure. It enables access to services to be restricted to authorized users.

The solution can be based on the IPsec security standard, in which case it is transparent to IP-based applications (such as video players). Alternatively, the solution can be based on the application using SRTP, ISMACryp, or MPEG2 TS CRYPT. This allows direct storage of the content in encrypted form. Additional application-specific content protection mechanisms could be freely combined with this solution, if desired, on top of the protection provided by these technologies.

Additional application-specific content protection mechanisms could be freely combined with this solution, if desired, on top of the transport layer.

OMA DRM 2.0 is used as the default framework for rights management. In its most common form, OMA DRM 2.0 manages the rights to use files stored in a device; this standard extends that to the case of receiving streaming content over a broadcast channel. It also provides a means of performing rights management over a broadcast channel.

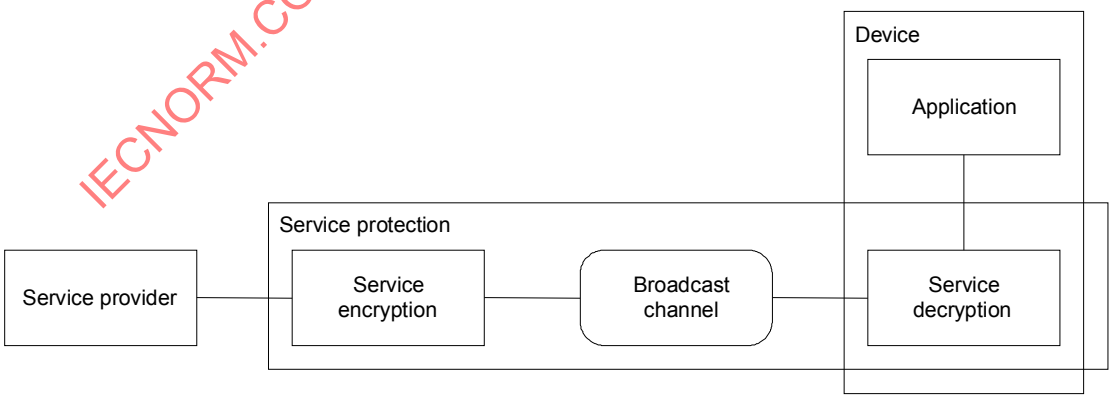


Figure 1 – System overview

Figure 1 shows the position of the service protection system within the overall architecture. IPsec allows the solution to be completely independent of the content format, while SRTP, ISMACryp and MPEG2 TS CRYPT provide alternatives for protecting content at the transport layer. Depending on the mode of operation, different normative prescriptions apply for the selection between the three ciphers. This is specified in the clauses about the supported systems.

4.2.2 Selected technologies

The following are the main standards on which the solution is based.

- Advanced Encryption Standard (AES, see FIPS PUB 197:2001) in the cipher block chaining mode, for actual content encryption. Furthermore, OMA DRM uses AES-WRAP in its rights objects and optionally AES CBC-MAC.
- Secure Internet Protocol (IPsec, see IETF RFC 4301) using the encapsulating security payload (ESP) protocol, for implementing service encryption and decryption as a function of the IP stack. Only the transport mode is used.
- Secure Real Time Protocol (SRTP, see IETF RFC 3711) as one option for implementing service protection at the transport layer. SRTP uses AES-CTR (counter mode).
- ISMA Encryption and Authentication, see ISMACryp 2.0, as a second option for implementing service protection at the transport layer. ISMACryp also uses AES-CTR (counter mode).
- The DVB-CSA as well as DES, 3DES, AES and M2 ciphers as the options for encrypting content in an MPEG2 TS.
- A traffic key delivery protocol and management as specified in this standard.
- Open Mobile Alliance (OMA) Digital Rights Management version 2.0 (OMA DRM 2.0, see OMA-ERP-DRM-V2_0) for managing rights to services, the associated service keys and the cryptographic protection of those keys themselves. This standard introduces some adaptations to OMA DRM 2.0 for IPDC service protection.
- Rights object delivery and device registration over a broadcast channel, without making any use of an interactivity channel, are also specified in this standard.
- Adapted Zero Message Broadcast according to Fiat Naor:1998 in 1-resilient mode.
- SimulCrypt (see ETSI TS 103 197) head-end interfaces.

The reasons for choosing these particular technologies as the basis of the solution include the following.

- AES is an efficient symmetric encryption method and an open standard that has hardware implementations. AES has many existing applications.
- IPsec/ESP is the standard way of keeping service decryption in receiving devices within the IP stack, invisible to the receiving applications, which thus remain independent of service protection and the carriers of the IP flows.

  NOTE An IPDC specific broadcast channel is only one means to transmit such IP flows. IPsec/ESP has many existing applications.
- SRTP is a standard way of performing service decryption at receiving devices within the transport layer. It can be used to carry all common forms of streaming content, which can be stored by a device for later decryption, if required.
- ISMACryp is also a standard way of performing service decryption within the transport layer. It provides end-to-end protection for transport and storage of streaming content.
- DVB-CSA is a widely deployed, standard way of encrypting content in an MPEG2 TS. The descramblers of most video processors implement the alternative DES, 3DES, AES and M2 ciphers.
- The traffic key (i.e., the actual content encryption key) management framework and protocol are specified in this standard. The efficiency and robustness of the solution is achieved by a particular delivery protocol and management scheme for the frequently changing traffic keys.
- OMA DRM2.0 is a state-of-the-art open standard for digital rights management. The use of OMA DRM2.0 allows deployments to the leverage existing implementations in the area of DRM-enabled services.

- OMA DRM 2.0, however, uses interaction over a two-way communication channel for device registration and guaranteed rights delivery. To adapt to broadcast devices, and to optimize the use of the broadcast channel, some new standardization is introduced.
- The main and foremost consideration for the system is network bandwidth efficiency, while maintaining realistic CPU requirements for the local device. By using zero message broadcast encryption, the network does not need to transmit keys to devices after registration. By choosing Fiat Naor, there is a good balance between network bandwidth consumption and local device processing requirements.

  The chosen group size is relatively small (with 256 or 512 devices), and, depending on the group size(m), the following is needed:

  – device transmission and storage for 2log(m) subscriber group keys (a.k.a. 'SGKs');

  – device computing power to derive (m-1) leaf keys;

  – device computing power to process, worst case, (m-1) leaf keys to create the "IEK" key that covers the SEK.

  Given the small size of the subscriber group, local device processing requirements are not considered a problem.

  Fiat Naor is used in 1-resilient mode, thereby making a collusion attack in theory possible. Making the scheme k-resilient, as discussed in Fiat Naor:1998 will increase the number of keys dramatically. The potential threat of 1-resilience is countered by measures described in 5.5.3.
- By using SimulCrypt interfaces for the T/C/S head-end, the 18Crypt solution can be easily integrated into existing DVB head-end configurations by using widely available SimulCrypt components.

4.2.3 Overview of four-layer model for service protection

This subclause introduces the four-layer model for service protection. The four-layer model is specified in 5.4. The four layers are specified in Clauses 6, 7, 8 and 9, respectively.

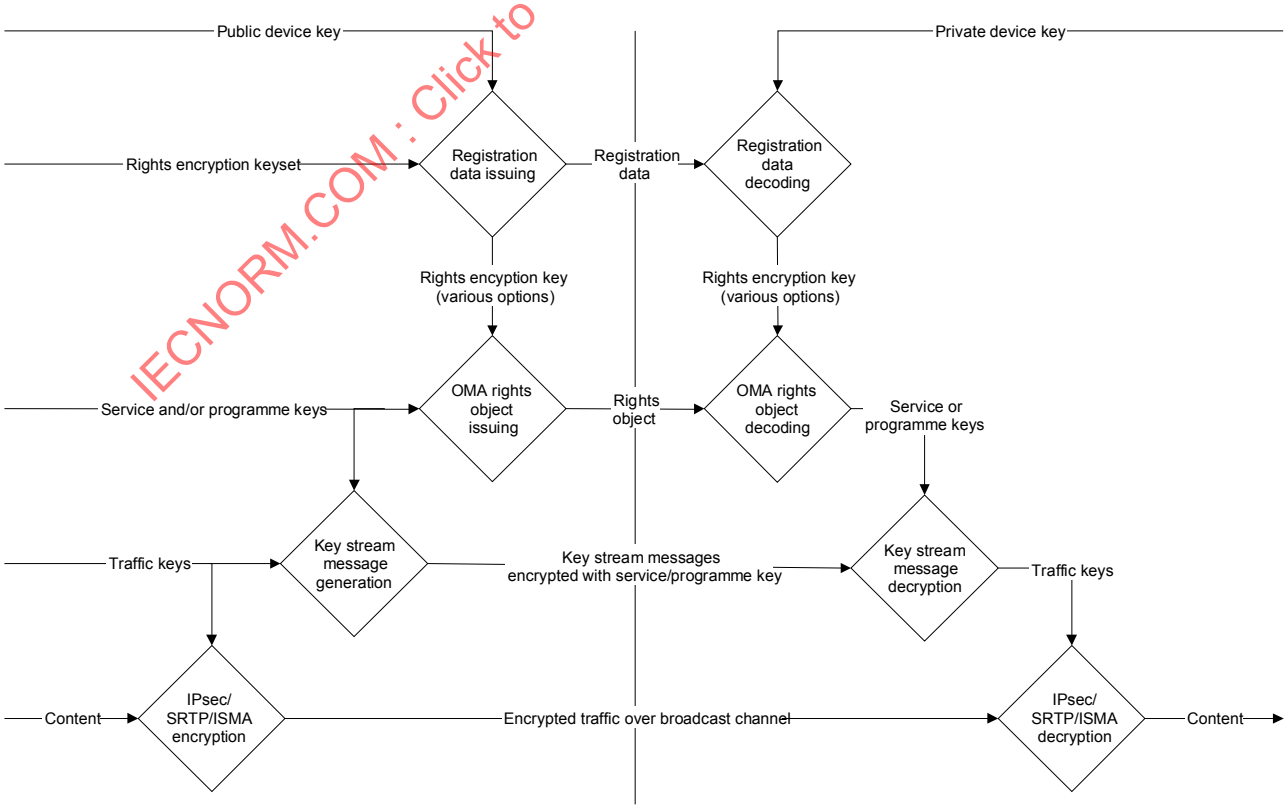


Figure 2 – Service protection via four-layer model

As illustrated in Figure 2, the solution is based on a four-layer cryptographic architecture, with an optional optimization to provide both secure subscription and pay-per-view purchase options for a single service. Actual service encryption is carried out according to AES using 128-bit symmetric traffic keys.

Traffic keys are applied:

- as part of standard IPsec security associations (SAs); or
- as an SRTP master key, from which the session key is derived according to the SRTP specification; or
- as an ISMACrypKey, from which the session key is derived according to the ISMACryp specification; or
- as a content key for MPEG2 TS CRYPT.

These are used by the IPsec, SRTP, ISMACryp or MPEG2 TS CRYPT layers to perform decryption automatically before passing the packets to the receiving application. The traffic keys are not protected by IPsec, but instead encrypted with a service or programme key on the key stream layer, above the IP socket interface. These broadcast messages carrying traffic keys are called key stream messages.

Key stream messages can contain two levels of encryption. Separate programme and service keys have different lifetimes and can be used to provide, for a single service, different granularities of purchase periods to different users. This allows for the efficient implementation of both subscription and pay-per-view business models for the same service. Pay-per-view customers are provided with a programme key that is only valid for a single programme while subscribers are given a service key, valid for reception of the service for some longer period. Within the key stream message, the traffic key is encrypted with a programme key, and the programme key is also carried, encrypted with the service key. Thus, pay-per-view customers can directly decrypt the traffic key, while subscribers can decrypt the programme key using the service key, which can then be used to decrypt the traffic key.

Key stream messages contain extensions to content IDs, which are carried in the service guide, for the programme and/or service. Devices use this ID to identify which rights object contains the keys to use for key stream message decryption.

Where the two-level service and programme functionality is not required, the traffic key can be directly encrypted with either the service or programme key with the service-key-encrypted programme key omitted.

The service or programme key(s) are transmitted to each receiving device within OMA DRM 2.0 rights objects (ROs). Such transmission of ROs can be done in two different ways, depending on whether the receiving device can make use of a separate interactivity channel:

- via a broadcast channel; or
- by using the separate interactivity channel.

In both cases, the ROs can be utilized by the customer device only, since the service or programme key sections are protected according to the OMA DRM 2.0 standard, or, in the broadcast case, by the variant of OMA DRM 2.0 specified in this standard.

When delivering rights objects over a broadcast channel, bandwidth is a major constraint. This standard addresses this problem in two complementary ways. Firstly, a new binary form of the OMA DRM 2.0 rights object, called a broadcast rights object (BCRO), is defined. Secondly, a method is specified for securely delivering BCROs to groups of devices using a single broadcast message. Valuable portions of rights objects are protected by group or unit keys, and when necessary, zero message broadcast encryption can be used to allow messages to be decrypted only by arbitrary sets of devices within a larger group.

An additional mechanism is available, as in OMA DRM 2.0, for rights objects to be issued to a group of devices known as a domain. It is expected that a domain will contain a number of devices belonging to the same user, and will be used by rights issuers to sell subscriptions allowing all devices within the domain to receive protected services.

Registration can be performed either via an interactivity or via a broadcast channel. In the case that an interactivity channel is used, the registration protocol is according to OMA DRM 2.0 and unit keys are delivered, protected with the public key of the device. This standard defines an efficient and user-friendly process for the registration of devices that do not support an interactivity channel, and a new protocol, called the 1-pass ROAP, is defined for the delivery of unit, group and broadcast keys via a broadcast channel.

The service key protection thus is based, according to OMA DRM 2.0, on a public key cryptosystem where the public key of the customer device is registered at each rights issuer and the corresponding private key is kept within the customer device.

4.3 End-to-end system

This subclause briefly specifies the major roles within the system. For more details, see 5.1.

Figure 3 shows a simplified view of the major actors in the system and their relationship to each other.

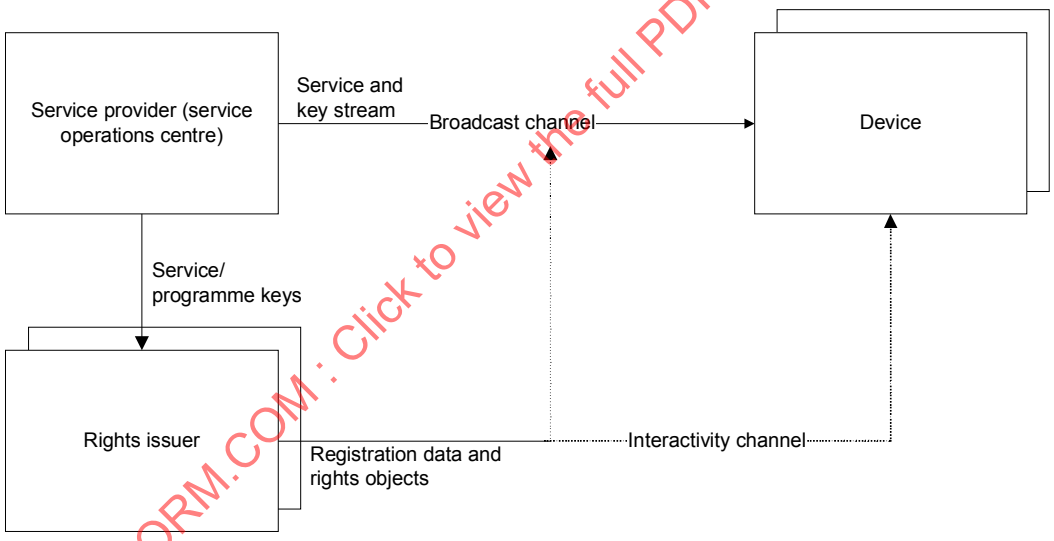


Figure 3 – Highly simplified view of the end-to-end system

The main actors are as follows.

- The service provider, also known as the service operations centre, broadcasts and encrypts the service and the key stream. It provides service and programme keys to the rights issuers.
- The rights issuer registers devices and provides rights objects to registered devices allowing them to decrypt the services that they are entitled to receive.
- The device receives the service, decrypts it (if it has the necessary rights objects) and presents it to the user.

4.4 Supported systems and device types

This standard supports stream based service access for the following systems.

- IP datacast over DVB-H, referred to in this standard as ‘IPDC over DVB-H system’.

- Digital broadcast system according to DVB-T, DVB-C or DVB-S standard, referred to in this standard as 'DVB T/C/S system'.
- Two versions of IPTV:
  - MPEG2 TS-based IP system;
  - non-MPEG2 TS-based IP system.

Based on the communication channels used, this standard supports the following device types, as listed in Table 1.

Table 1 – Supported systems and device types

System	IPDC over DVB-H			DVB T/C/S		IP	
						MPEG2 TS-based	Non-MPEG2 TS-based
See	Clause 13 Subclause B.1.1			Clause 14 Subclause B.1.2		Clause 15 Subclause B.1.3	Clause 16 Subclause B.1.3
	Broadcast device	Mixed-mode device	Interactive mode device	Broadcast device	Mixed-mode device	IPTV device	IPTV device
Comm. channel(s)	IPDC	IPDC + IC	IC	DVB-T/C/S	DVB-T/C/S + (optional) IC	IC	IC
AV content (Traffic layer, Clause 6)	Via SRTP or IPsec over IP	Via SRTP or IPsec over IP	Via SRTP or IPsec over IP	Via PES over MPEG2 (MP) TS	Via PES over MPEG2 (MP) TS	MPEG2 AV data via PES over IP	Via SRTP, IPsec or ISMACryp over IP
TEK (Key stream layer, Clause 7)	KSM via IP (packets)	KSM via IP (packets)	KSM via IP (packets)	KSM via SI table (section) in TS	KSM via SI table (section) in TS	KSM viaMPEG2 table (section) in TS	KSM via IP (packets)
RO (Rights manage-ment layer, Clause 8)	BCRO via IP packets, using RI services (Clause 10)	BCRO via IP packets over IPDC or ICRO via XML/ROAP over IP or OBEX	ICRO via XML/ROAP over IP or OBEX	BCRO via SI table (section) in TS, using RI services (Clause 10)	BCRO via SI table over TS or ICRO via XML/ROAP over IP or OBEX	ICRO via XML/ROAP over IP	ICRO via XML/ROAP over IP
Registration data (Regis-tration layer, Clause 9)	Via IP packets, using RI services (Clause 10)	Via IP packets over IPDC or via XML/ROAP over IP or OBEX	Via XML/ROAP over IP or OBEX	Via MPEG2 tables (sections) in TS, using RI services (Clause 10)	Via MPEG2 tables (sections) in TS or via XML/ROAP over IP or OBEX	Via XML/ROAP over IP	Via XML/ROAP over IP
Service guide (Clause 10)	IPDC over flute	IPDC over flute	IPDC over flute	Purchase and EPG data via extra SI tables plus DVB-SI or A65 in TS	Purchase and EPG data via extra SI tables plus DVB-SI or A65 in TS;op-tionally as additional programme information data over IC	DVB-IPI BCG	DVB-IPI BCG
NOTE DVB-T/C/S devices may use linkage descriptors for more options to signal where the messages will be carried, as explained in B.6.3. This has been omitted in this table for the sake of clearness.							

4.5 Service protection versus content protection

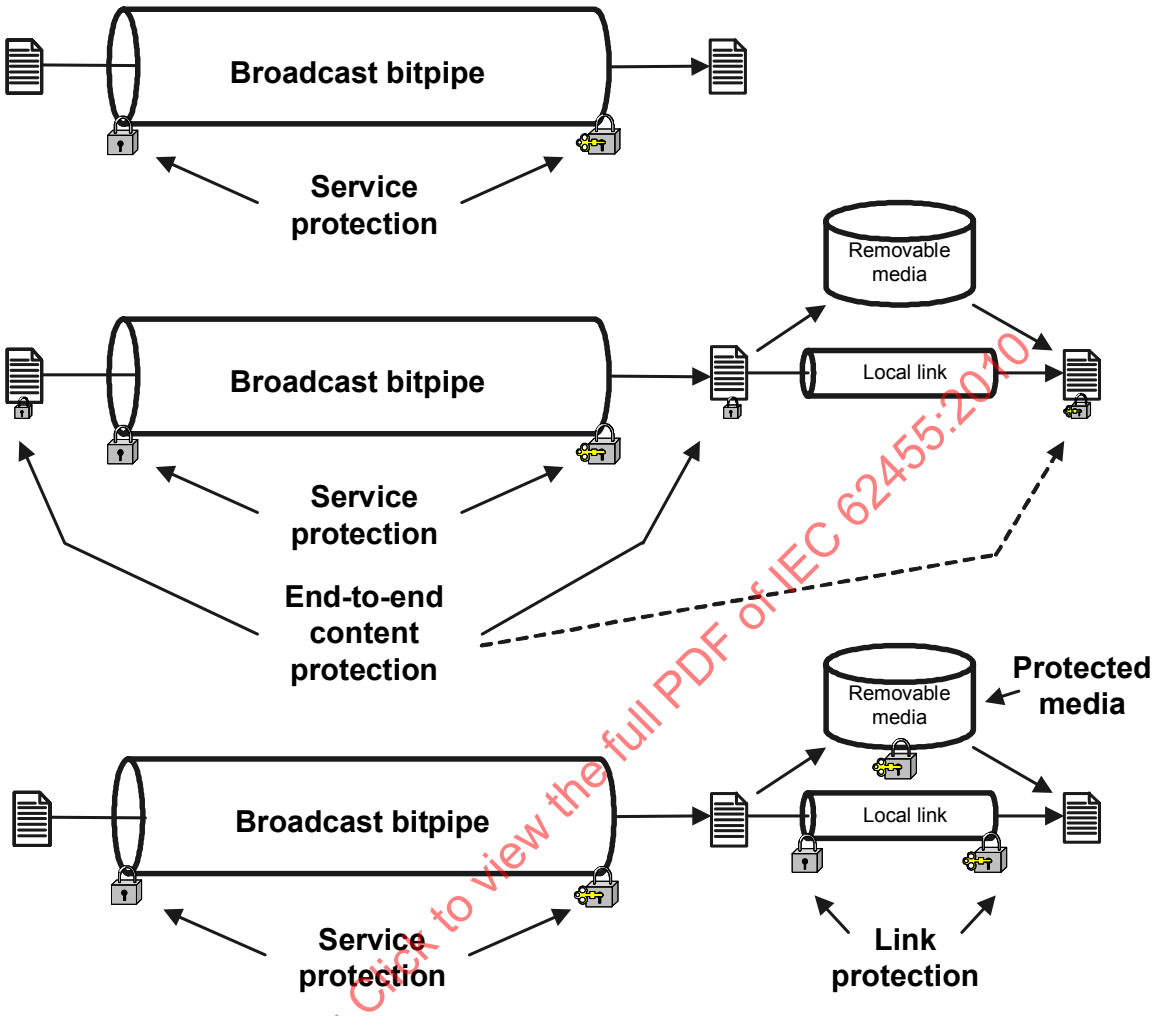


Figure 4 – Service protection versus content protection

Figure 4 illustrates the conceptual difference between service protection and content protection.

Service protection ensures that only those authorized to access the broadcast service can do so. Service protection is removed at reception time.

Content protection refers to protecting the content either throughout the content lifecycle (end-to-end content protection) or subsequent to delivery through the service protection system (post-delivery content protection).

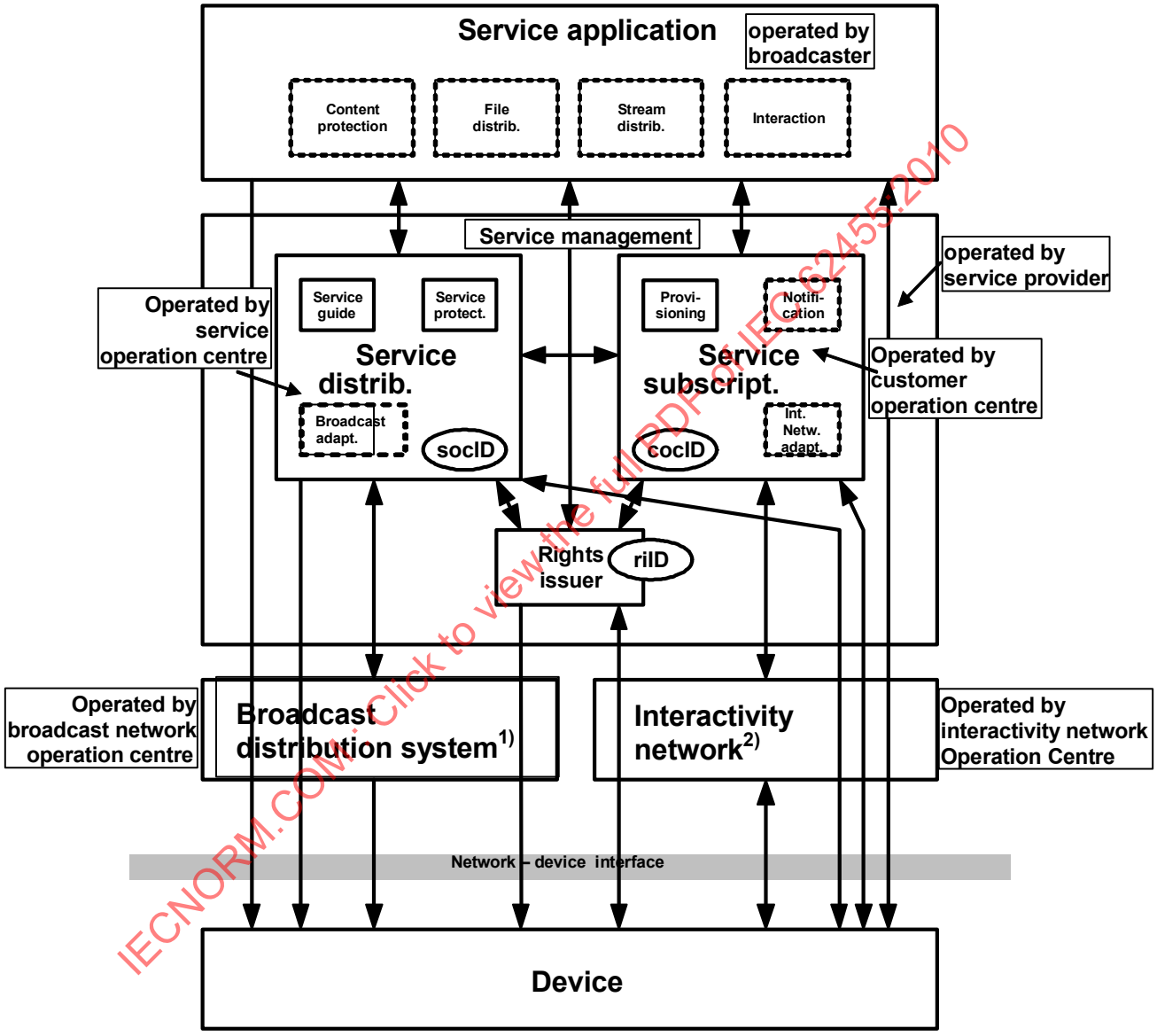
The services purchase and protection system defined in the present standard provides service protection, and it can optionally also be used to facilitate end-to-end content protection. The access permission in rights objects controls access to the broadcast service and allows immediate rendering of the content. In order to be able to play recorded content, play permission is required. If only service protection is needed, export permission can be used to allow exporting in the clear.

In some cases, for instance to facilitate sharing of the protected content with different devices, it could be desirable to export the content to a separate post-delivery content protection system. The usage state information needed by such a content protection system can be carried as a constraint for export permission.

5 General specifications

5.1 End-to-end architecture

This subclause specifies the end-to-end architecture of the system in a network centric way and illustrates how the public key infrastructure relates to the end-to-end architecture. The figures in this subclause can be used as a reference and overview when reading this standard.



NOTE 1 Broadcast distribution systems may be:

- IP datacast over DVB-H;
- digital broadcast system using the DVB-T, DVB-C or DVB-S standard;
- the Internet.

NOTE 2 Interactivity networks may be:

- cellular radio networks like GSM/GPRS;
- ADSL, ATM or DOCSIS networks;
- the Internet.

Figure 5 – Service protection and purchase entities and names (broadcast architecture)

Figure 5 names several entities used in this standard. The arrows in this figure represent information flow between the elements. All these entities should be considered as logical entities, i.e. actual deployment may be different and entities indicated separately in Figure 5 may well be merged into larger systems. One should also note that there may be additional connections and entities, such as the off-line communication of broadcast mode devices and certification authority needed for public key infrastructure.

The service application represents any content that can be sent over a broadcast channel. Besides digital television it can be, for example, discrete files, streams or interactive services. The service distribution management's role is to create and manage protected broadcasts as specified in this standard. Service subscription management takes care of the purchase of services. The rights issuer takes care of device registration and rights issuing.

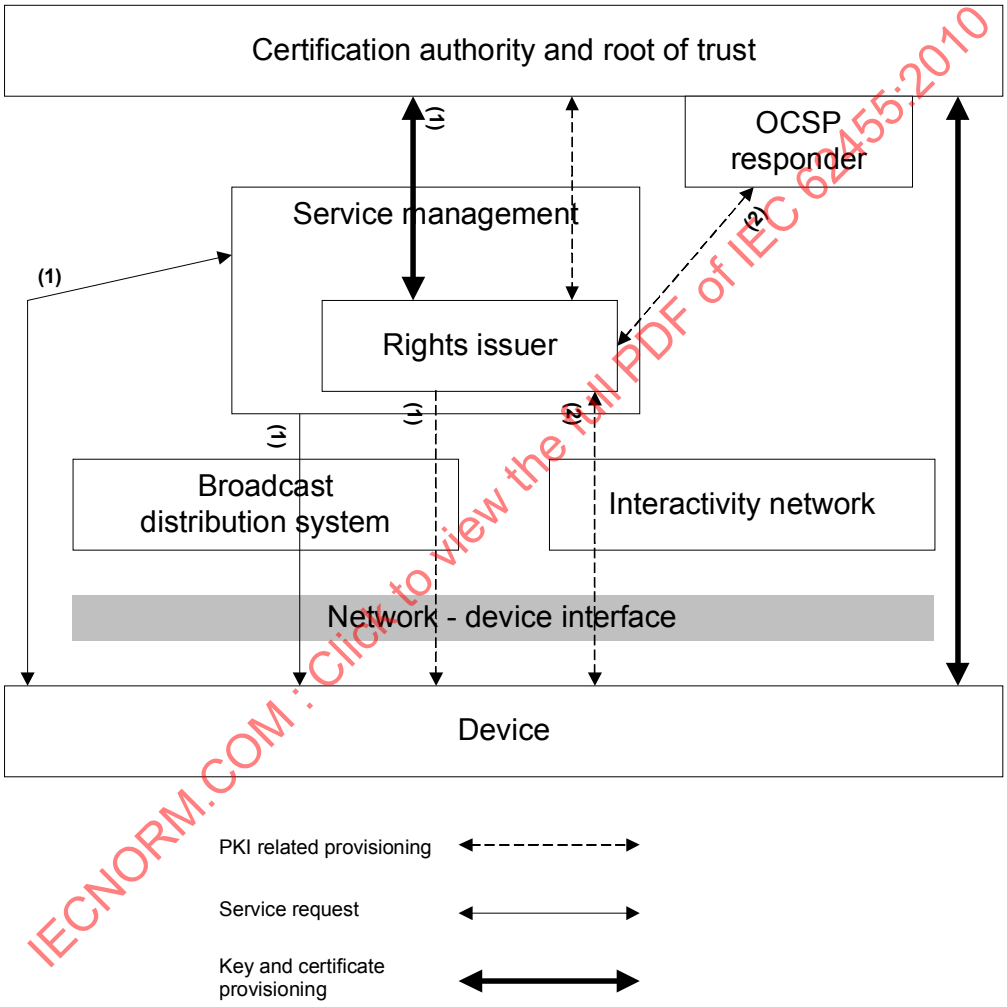


Figure 6 – Public key infrastructure

Figure 6 maps entities from the public key infrastructure to the broadcast architecture. A root certification authority provides keys and certificates to devices and rights issuers. If the device and rights issuer do not have the same root certification authority, the rights issuer cannot register the device or issue rights to the device.

Lines marked with (1) relate to the broadcast mode of operation and lines marked with (2) relate to the interactive mode of operation (compare Figure 6 with Figure 22 and Figure 37).

5.2 Special cases

5.2.1 Free-to-air services

Free-to-air services, by definition, are broadcast without any service protection. Any device can receive these services. In this subclause, ‘air’ may also mean ‘cable’, etc.

> Free-to-air services shall not be broadcast using IPsec, SRTP, ISMACryp or MPEG2 TS CRYPT. In this case, a key stream will not normally be broadcast, but if one is broadcast, it shall be ignored by the device.

Otherwise, free-to-air services are beyond the scope of this standard.

5.2.2 Free-to-view services

Free-to-view services are broadcast with service protection, but no charge is made to receive the services. However, reception can be restricted to certain users.

It is expected that free-to-view services will be implemented by either

- requiring users to "subscribe" to the service in the normal way, although no charge is made for the subscription (rights objects are delivered to "subscribers"); or
- distributing rights objects to all devices in a population or to particular pre-existing subscriber groups or registered devices without requiring a specific "subscription".

However, free-to-view services may be implemented in any way that is compliant with this standard.

5.3 Service guide and purchase

Broadcast systems may have some form of service guide available. This service guide is a source of service, programme and content information. Service and content discovery may then be based on information transmitted in the service guide.

In systems supporting automated purchase of services and content, it is assumed that a service guide is available, which also contains necessary information for making a purchase of a service or programme. This means that for each service and programme, the service guide contains all the necessary information for making a purchase and for the device to find services and programmes.

NOTE ”Service guide” is used as a generic term in this standard. Specific systems may use specific terms (for example, ESG in IPDC over DVB-H systems; see 13.6).

Figure 7 illustrates a high-level overview of a service guide and purchase system. Line (1) in Figure 7 represents the broadcast of a service guide over a broadcast channel. Lines (4) and (5) represent purchase and lines (2) and (3) represent the delivery of necessary rights objects.

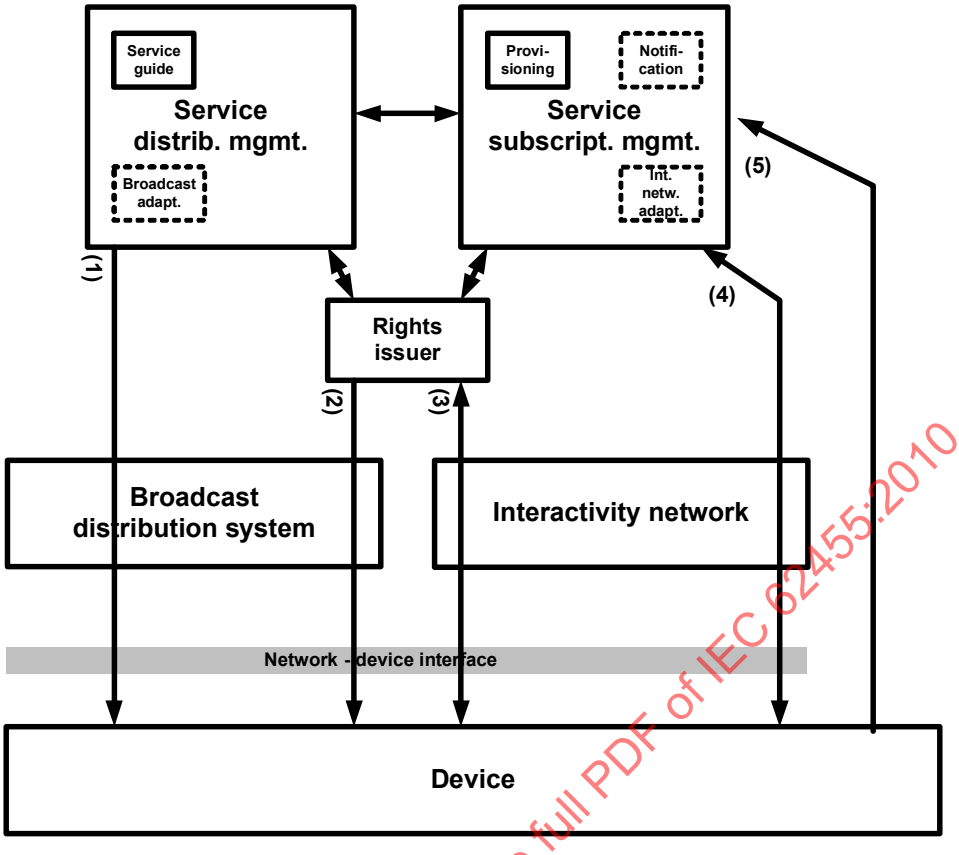


Figure 7 – Overview of service guide and purchase

Service guide data may be carried over a broadcast or interactivity channel. To be able to use the protected services and content specified in this standard, a device needs to be able to receive and parse service guide data. Unless the service or content is free to air (see 5.2.1), the user needs to make a purchase and the service guide contains all the needed information to do so. Interactive and mixed-mode devices can find the URI and other necessary parameters from the service guide. For broadcast devices, the service guide contains a ServiceID (see B.2.6.3) that identifies a purchased service or content. Line (4) in Figure 7 represents purchase by interactive and mixed-mode devices over an interactivity channel and line (5) represents off-line purchase by broadcast devices.

With interactive and mixed-mode devices, a purchase initiates the sending of an ICRO over an interactivity channel; line (3) in Figure 7. With broadcast and mixed-mode devices, BCROs are sent over a broadcast channel. Before purchases can happen, devices need to be registered with the relevant rights issuer; see Clause 9.

For more information on service guides, see Clause 10 for general specifications and 13.6, 14.6, 15.7 and 16.6 for system specific specifications.

5.4 Four-layer model – Key hierarchy

5.4.1 General

A four-layer key hierarchy is used to implement service protection. This four-layer model is analogous to the one used for protection in DVB systems. The four layers are as follows.

a) Traffic layer (content encryption; DVB equivalent is the CSA layer).

b) Key stream layer (DVB equivalent is the ECM layer).

c) Rights management layer (DVB equivalent is the EMM layer).

d) Registration layer.

The information in the traffic layer and the key stream layer is transmitted over a broadcast channel. The information in the rights management and registration layer may be distributed over a broadcast channel and/or an interactivity channel. The remainder of this subclause specifies the key hierarchy. Specific specifications of each of the layers can be found in Clauses 6, 7, 8 and 9, respectively. The way these layers are used in specific systems is specified in Clauses 13, 14, 15 and 16.

5.4.2 Keys on the traffic layer

On the lowest level, the data shall be encrypted using one of IPsec, SRTP, ISMACryp, or MPEG2 TS CRYPT, depending on the application. This layer is called the traffic layer. The key used to encrypt the traffic on this layer is called traffic encryption key, or TEK. The TEK changes frequently in the order of once per minute to once per second.

The encryption of the content can be written as

$$E\{TEK\}(C)$$

with C being the data and E{TEK}() being the encryption function using the key TEK.

The data can be recovered by using the decryption function D() with the same key TEK:

$$C = D\{TEK\}(E\{TEK\}(C))$$

As a symmetric encryption is used to encrypt the data, the data is recovered using the same key.

5.4.3 Keys on the key stream layer

5.4.3.1 General

The TEK itself is transmitted on the key stream layer. The TEK shall be encrypted using either a programme encryption key (PEK) or a service encryption key (SEK). The use of two different keys to protect the TEK allows for the models specified in the following three subclauses to be used.

5.4.3.2 Service-based subscription

If the service is made available to customers by subscription only, the following applies.

- If access rights change per programme, a programme key is used within the key stream message but is never delivered separately in a rights object. The scheme specified in 5.4.3.3 shall be used.
- If access rights do not change per programme, a programme key shall not be used and the scheme below shall be followed.

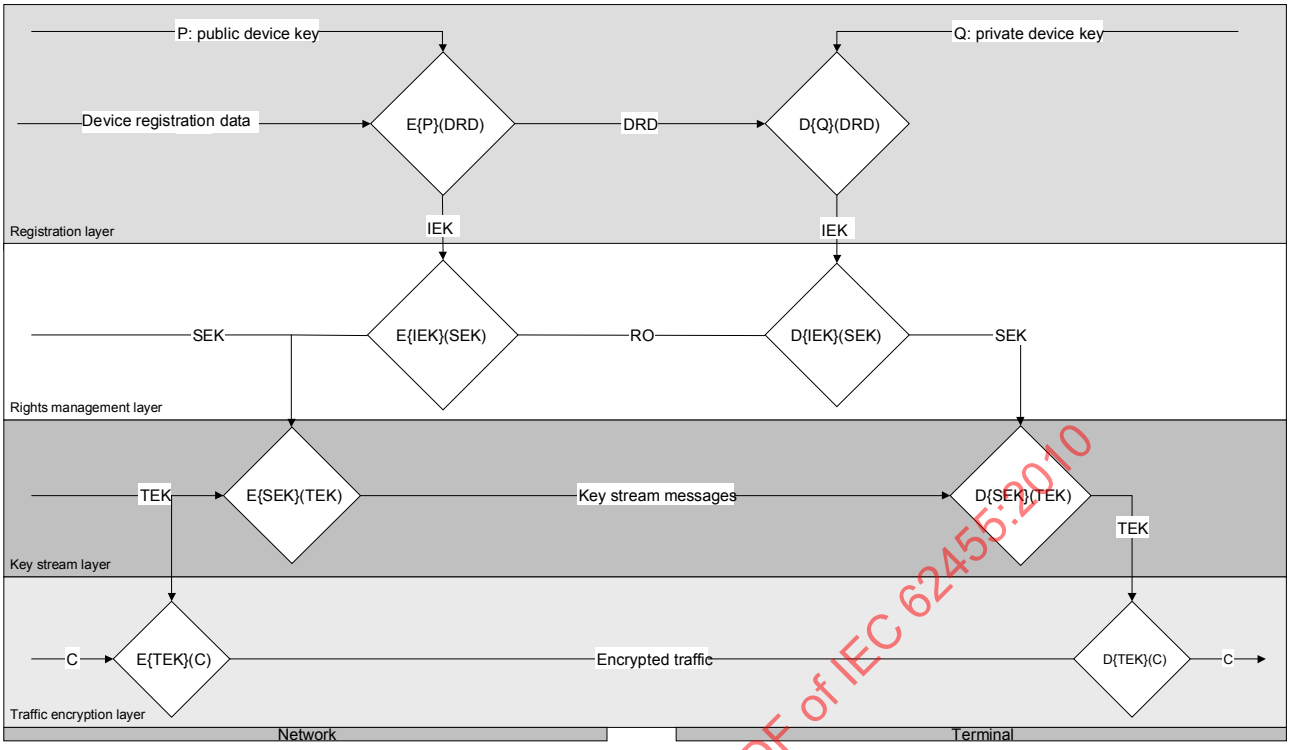
$$E\{SEK\}(TEK)$$

and

$$TEK = D\{SEK\}(E\{SEK\}(TEK))$$

The SEK is transmitted to devices as part of the rights objects on the rights management layer. These ROs shall be either normal OMA DRM 2.0 ROs in the case of an interactive device or BCROs for both mixed-mode and broadcast devices.

Figure 8 shows the key hierarchy for the case of a service-based subscription.



NOTE DRD – Device registration data: device-assigned keys during registration; see 5.4.6 and 5.4.7.

Figure 8 – 4-layer key hierarchy – Use of SEK only

5.4.3.3 Pay-per-view-based purchase and service-based subscription

If content is made available both via a service subscription and via a pay-per-view-based purchase then the TEK shall be encrypted with the PEK:

$$E\{PEK\}(TEK)$$

Devices that do not have a service-based subscription to that service can acquire the entitlement for a specific pay-per-view event. The RO for that pay-per-view event shall contain a PEK. This PEK can be used to decrypt the TEK:

$$TEK = D\{PEK\}(E\{PEK\}(TEK))$$

To allow devices with a service-based subscription to access the service as well, the PEK encrypted with the SEK is also carried in the key stream message. Therefore, the KSM carries

$$E\{SEK\}(PEK)$$

and

$$E\{PEK\}(TEK)$$

In order to decrypt the TEK given only the SEK, the device has to do the following decryption:

$$TEK = D\{PEK\}(E\{PEK\}(TEK))$$

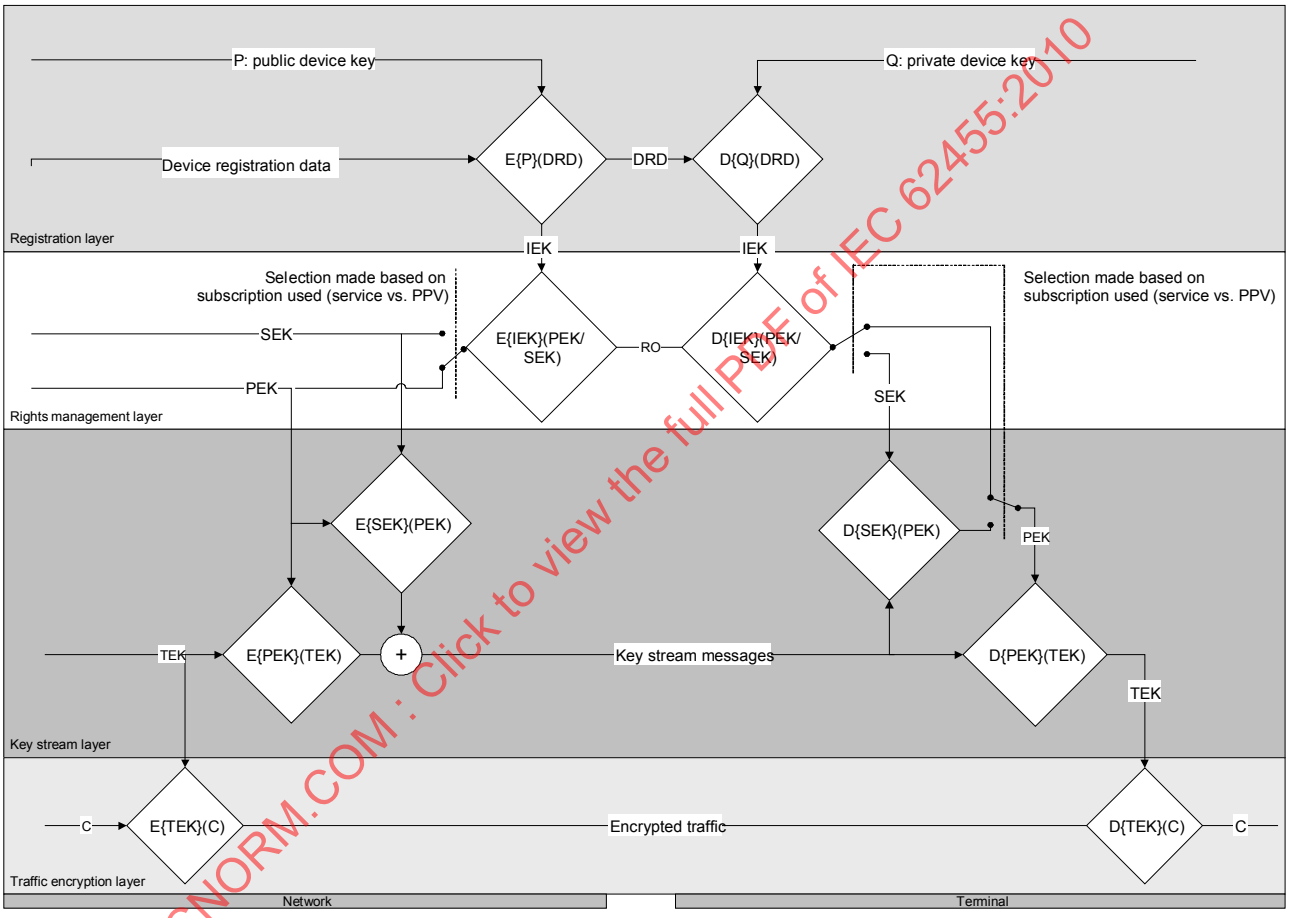
with

$$PEK = D\{SEK\}(E\{SEK\}(PEK))$$

hence

$$TEK = D\{D\{SEK\}(E\{SEK\}(PEK))\}(E\{PEK\}(TEK))$$

The lifetime of a PEK is expected to last only for the duration of a specific pay-per-view event while the SEK is expected to last for a longer period.



NOTE DRD – Device registration data: device-assigned keys during registration; see 5.4.6 and 5.4.7.

Figure 9 – 4-layer key hierarchy – Use of PEK and SEK

Figure 9 shows the four-layer key hierarchy in the case of service subscription and pay-per-view.

5.4.3.4 Pay-per-view-based consumption

If content can be consumed on a pay-per-view-only basis, that means that the content is not available via a service subscription, then the TEK may be encrypted with a PEK or the TEK may be encrypted with a SEK. Both PEK and SEK are interchangeable in this case.

5.4.4 Keys on the rights management layer (interactive mode)

If a rights object containing the SEK or PEK is delivered via an interactivity channel (see A.8.3.2), the rights encryption key is used to decrypt the SEK/PEK. OMA-ERP-DRM-V2_0 specifies how a device can obtain and use a rights encryption key. If the RO contains a CEK, it shall be processed according to OMA-ERP-DRM-V2_0 and is beyond the scope of this standard.

5.4.5 Keys on the rights management layer (broadcast mode)

In broadcast mode, the SEK and PEK are transmitted to the device on the rights management layer as part of a SEAK or PEAK in a BCRO, see A.8.3.3.

The keys used to encrypt and decrypt the SEK or PEK depend on the addressing mode of the BCRO (see 5.5.2). The term "inferred encryption key" (IEK) is used to indicate the key used to en-/decrypt the SEK/PEK without specifying which exact key is used.

- **RO addressed to a unique device**

  In the case where an RO is addressed to a unique device, the IEK used to encrypt the SEK or PEK shall be derived from the unique device key (UDK), as specified in 5.4.7, which UDK was delivered during device registration.

- **RO addressed to a subscriber group (subset of unique group)**

  In the case where an RO is addressed to a subset of a unique group (subscriber group), the IEK shall be deduced from the subscriber group keys (SGKs) by use of zero message broadcast encryption. See 5.5.3 for an explanation of the zero message broadcast encryption concept. See Clause A.7 for an explanation of how the IEK is deduced given the bit access mask and the SGKs.

- **RO addressed to a unique group**

  In the case where an RO is addressed to all devices in a unique group, the IEK used to encrypt the SEK or PEK shall be derived from the unique group key (UGK), as specified in 5.4.7.

- **RO addressed to a domain**

  In the case where an RO is addressed to a domain, the IEK used to encrypt the SEK or PEK shall be derived from the broadcast domain key (BDK), as specified in 5.4.7, which BDK was delivered during device registration.

- **RO containing a CEK**

  In the case where an RO is for an OMA DRM 2.0 content format (for example, a DCF), the asset shall carry a CEK object and an additional cipher value. Decryption of the key material is defined by OMA-ERP-DRM-V2_0.

5.4.6 Keys on the registration layer (interactive mode)

This is according to OMA-ERP-DRM-V2_0 and is beyond the scope of this standard.

5.4.7 Keys on the registration layer (broadcast mode)

The registration layer delivers the keys used for the broadcast mode of operation. There are several keys used for authentication and decryption purposes.

Table 2 – Keyset in the registration data

Name of key	Description	Remark
UGK	unique_group_key	
SGK	subscriber_group_key	Used for zero message broadcast
BDK	broadcast_domain_key	Used for domains
UDK	unique_device_key	
RIAK	ri_authentication_key	Used for authentication
UDF	unique_device_filter	Eurocrypt address, not a key
SBDF	shortform_broadcast_domain_filter	domain_id address, not a key
LBDF	longform_broadcast_domain_filter	domain_id address, not a key
TDK	token_delivery_key	Used for token delivery, not for registration

The keyset itself is delivered to the device in a protected format, as part of the device registration data (see 9.3.2.7.2 for details).

The RI generates a session key (SK) to protect the keyset_block (UGK, SGK1..n, UDK, BDK, RIAK, UDF, SBDF, LBDF and/or TDK), which carries the keyset specified in Table 2 above.

$$\text{encrypted\_keyset\_block} = E\{SK\}(\text{keyset\_block})$$

The RI encrypts the SK and the encrypted_keyset_block (together called the SK + encrypted_keyset_block) into a sessionkey_block, such that

$$\text{sessionkey\_block} = E\{DP\}(SK + \text{encrypted\_keyset\_block})$$

where the sessionkey_block is encrypted with the public key of the device (DP).

If the keyset_block does not fit into the size of the sessionkey_block, the remainder is kept as surplus_block. See 9.3.2.7.3.2 for details.

The complete message (header, sessionkey_block and optional surplus_block) is protected by a single source authenticity check, such that:

$$\text{signature\_block} = A\{RIQ\}(\text{message})$$

where the RIQ is the private key of the RI.

Upon reception, the device follows the rules specified above in reverse order:

$$V\{RIP\}(\text{signature\_block})$$

$$SK + \text{encrypted\_keyset\_block} = D\{DQ\}(\text{sessionkey\_block})$$

$$\text{keyset\_block} = D\{SK\}(\text{encypted\_keyset\_block})$$

where the signature_block is verified with the RI public key (RIP).

The encrypted sessionkey_block contains the session key (SK) plus encrypted_keyset_block (together called the SK + encrypted_keyset_block) and is decrypted with the private key of the device (DQ).

If the surplus_block is present, it is concatenated to the keyset_block from the sessionkey_block. See 9.3.2.7.2.2 for details.

The encrypted_keyset_block, decrypted with the session key (SK), produces the keyset_block, containing the keyset (UGK, SGK, UDK, RIAK, UDF), which never leaves the DRM agent.

The notation HMAC_SHA1{k}(s) is used to denote the computation of HMAC, see IETF RFC 2104, with SHA1, see FIPS PUB 180-2:2002, as the hash function keyed by the key 'k' over the string 's'. HMAC_SHA1_128{k}(s) is used to denote the 128 most significant bits of HMAC_SHA1{k}(s) output.

A key IEK is "derived" from the UGK, SGK, UDK or BDK to decrypt the BCRO, such that

$$IEK = HMAC\_SHA1\_128\{UGK\}(salt)$$

or

$$IEK = HMAC\_SHA1\_128\{NK_i \,||\, ... \,||\, NK_j\}(salt)$$

where the NKs are NK keys ordered according to the index (such that i < j) that are required for creating the key for the desired group. The keys NK are obtained using the scheme specified in 5.5.3. See 5.5.3 and A.7.

or

$$IEK = HMAC\_SHA1\_128\{UDK\}(salt)$$

or

$$IEK = HMAC\_SHA1\_128\{BDK\}(salt)$$

The IEK is used to decrypt the part of the BCRO containing key material, which carries CEK or SEAK and/or PEAK, see A.8.3.3. The 'salt' parameter is the BCI value in the asset structure of the BCRO. The BCI value from the first asset structure in a BCRO shall be used for all assets in a BCRO structure.

5.4.8 Authentication overview

5.4.8.1 Authentication hierarchy

Figure 10 explains the authentication "hierarchy" of the system.

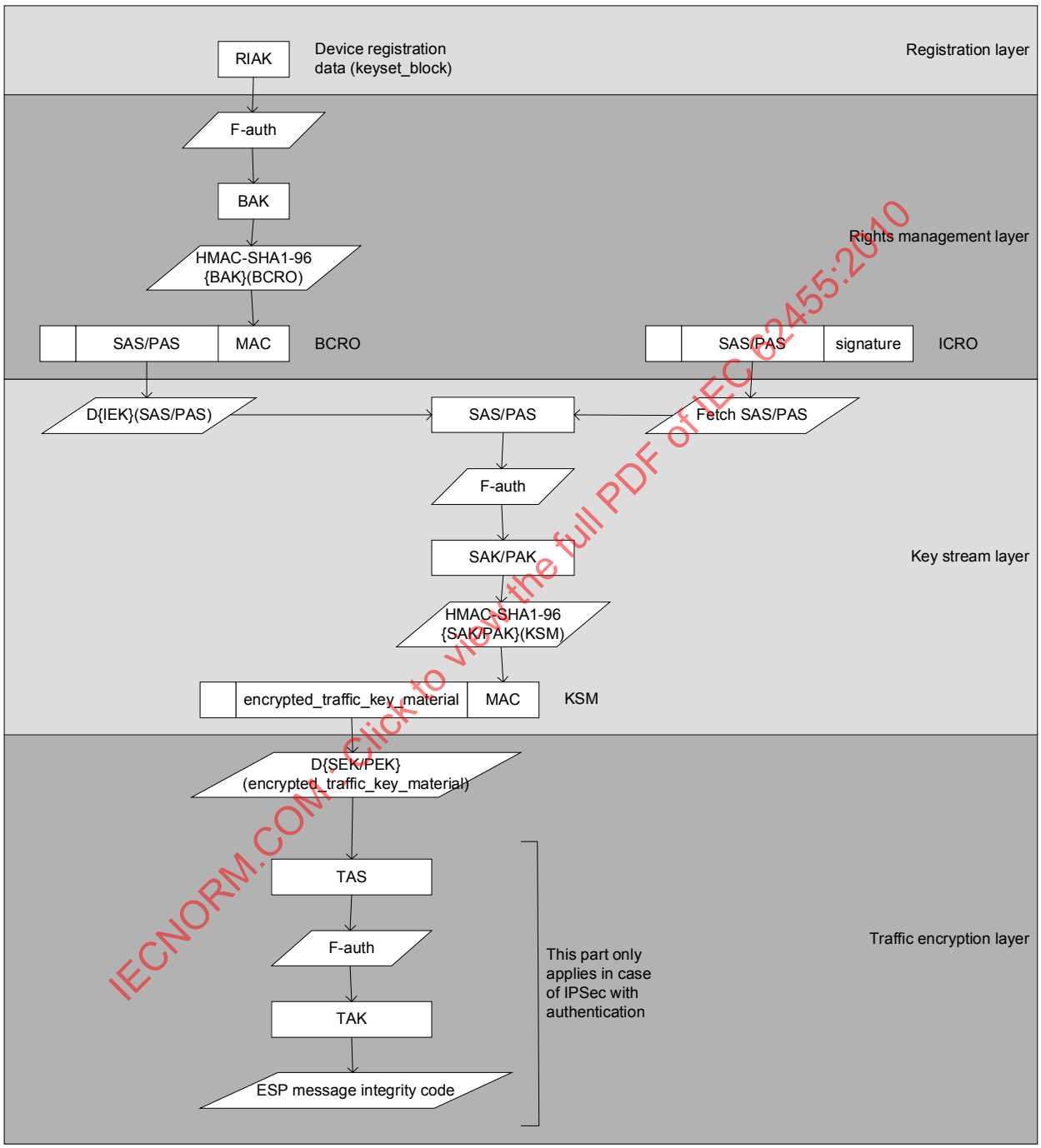


Figure 10 – Authentication hierarchy

5.4.8.2 Authentication keys on the traffic layer

When IPsec is used with authentication, the message shall be verifiable by the ESP integrity code. This shall be done by means of the traffic authentication key (TAK), which is derived from the traffic authentication seed (TAS) (see A.8.2).

5.4.8.3 Authentication keys on the key stream layer

The KSM shall be authenticated and the integrity of the message shall be verified. This shall be done by means of the programme authentication key (PAK) and/or the service authentication key (SAK), which are derived from the programme authentication seed (PAS) and the service authentication seed (SAS), which are delivered as part of the RO (see A.8.3).

5.4.8.4 Authentication keys on the rights management layer (broadcast mode)

The BCRO shall be authenticated and the integrity of the message shall be verified. This shall be done by means of the BCRO authentication key (BAK), which is derived from the RI authentication key (RIAK), which is delivered during registration (see A.8.4).

5.4.8.5 Authentication keys on the registration layer (broadcast mode)

The RI authentication key (RIAK) shall be delivered during registration as part of the device_registration_response(). See 9.3.2.4 for details.

5.5 Deployment for broadcast mode of operation

5.5.1 Concept of Domains – Interactive and broadcast domains

Content in OMA DRM 2.0 can be bound to a device or to a domain; see OMA-ERP-DRM-V2_0. In this standard, we refer to a domain as specified by OMA-ERP-DRM-V2_0 as an interactive domain.

A domain in OMA-ERP-DRM-V2_0 is a group of one or more devices that share a common secret, the so-called domain key. In the case of content that is bound to a domain, the content encryption key of that content as stored in the RO associated with that content is encrypted with the domain key of that domain. A device that receives an RO that is bound to a domain of which it is a member can freely pass that RO to other devices belonging to the same domain. The other devices in the same domain can then access that content as allowed by the RO.

OMA-ERP-DRM-V2_0 defines protocols with which a device can join and leave a domain: the ROAP join domain protocol and the ROAP leave domain protocol. In order to use domain keys, the device shall have joined the corresponding domain. The equivalents for these protocols in the case that there is no interactivity channel are defined in 9.3.3.

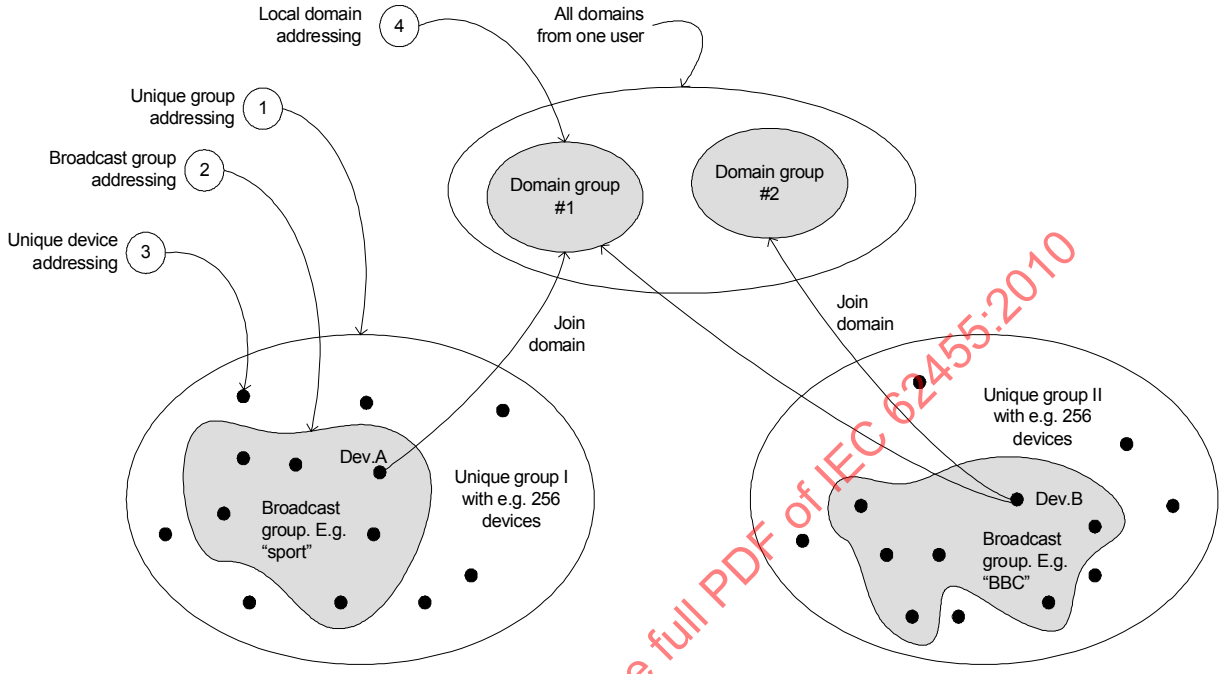
The equivalent of the interactive domain in case there is no interactivity channel is the broadcast domain. In the BCRO, there is a facility to indicate that the BCRO is intended for a broadcast domain, using the address_mode. If the address_mode is set to "domain", the domain key is used for the encryption of the key material in the BCRO. Furthermore, if the address_mode is set to "domain", the domain_id is created by concatenating the address_field with the domain_id_extension.

NOTE While the domain ID and domain key may be the same in the case of the interactive and broadcast domain, the content for broadcast domains with BCROs is not interoperable with content for interactive domains with ICROs.

5.5.2 Addressing (group/subset/device/domain)

5.5.2.1 Concept of addressing

The registration data supports four methods of addressing devices, as explained in Figure 11.



Key

1 Addressing a unique group – use unique group key (UGK)

2 Addressing a subscriber group – use subscriber group key(s) (SGK)

3 Addressing only one device – use unique device key (UDK)

4 Addressing a broadcast domain – use broadcast domain key (BDK)

Figure 11 – Explaining the concept of addressing

A unique group contains all the devices in a group. A subscriber group can be smaller than, or as large as, the unique group. Alternatively, a device can be addressed via a (broadcast) domain group. A unique device can be part of a unique group and/or a subscriber group and/or a broadcast domain. One or more unique groups form the population of devices. In this example, the group size is 256 devices. Group sizes of 256 and 512 are supported, as these are acceptable group sizes when it should be needed to revoke devices. Using a larger group size is not supported because of the fact that eventual revocation of such a group easily concerns too many devices.

The following subclauses specify the relationship between the registration data and the broadcast rights object. The registration data is sent to the device after successful registration of the device. At a later stage, the device can receive a BCRO as a means to obtain the content encryption key, which in turn is used to decrypt the encrypted AV content. The content key may carry the SEK and/or PEK. The RI encrypts the content key with an inferred encryption key (IEK). The following subclauses specify the different addressing modes, how the message is filtered and what type of IEK will be used by the device to obtain the content key.

A device supporting broadcast mode of operation shall support all four addressing modes. The network operator can choose to send registration data with a keyset of UGK and/or SGK and/or UDK and/or BDK to the device at registration time.

See 5.4 for the general overview of the process used to construct the IEK.

5.5.2.2 Addressing the unique group

To access the whole group, the following (oversimplified) BCRO is used (see Figure 12).

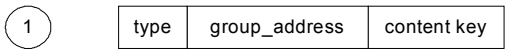


Figure 12 – (Oversimplified) group BCRO

- The group_address was delivered with the registration data and is used to filter for the message (see 9.3.2.7 for details).
- The content key (which can carry the SEK and/or PEK) is encrypted with an inferred entitlement key (IEK). In this case, the unique_group_key (UGK) is used to derive the IEK, see 5.4.7. The UGK used by the RI is identical to the key that was delivered with the registration data sent to the device (see 9.3.2.7 for details).
- All the devices in the group can use the content key.

5.5.2.3 Addressing a subscriber group

The subscriber group is a privileged subset of the unique group. Two methods are available to address the subscriber group, but either one can be used.

- At registration, the registration data delivers a set of subscriber_group_keys (SGK) to the Device. By using zero message broadcast encryption (see 5.5.3), an inferred encryption key (IEK) can be constructed from the SGKs sent to the device. The RI uses this IEK to encrypt the content key and only the devices with the matching set of SGKs on board can construct the same IEK.
- For reasons of completeness, it is mentioned that is also possible to deliver no SGKs with the registration data. With 0 (zero) SGKs, it is, of course, not possible to deduce an IEK. In this case, unique device addressing with a unique_device_key (UDK) to derive the IEK is used. See 5.5.2.4 for more details.

NOTE It is inefficient for large populations to use unique addressing instead of (broadcast) group addressing, since it quickly consumes a considerable amount of bandwidth.

To access the subscriber group, the following (oversimplified) BCRO is used (see Figure 13).

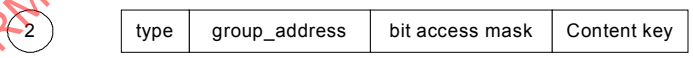


Figure 13 – (Oversimplified) subscriber group BCRO

For both methods, the group_address was delivered with the registration data and is used to filter for the message.

- The group_address is part of the unique_device_filter (UDF) address that was sent with the registration data to the device and is used to filter for the message (see 9.3.2.7 for details).
- The Eurocrypt-style bit access mask, see EN 50094:1992, prescribes which group members are "entitled" to use the content key (which can carry the SEK and/or PEK). This bit mask indicates which group members of the subscriber group can deduce the broadcast key from the zero message broadcast keys in the device.
- The device can construct the inferred encryption key (IEK) from the SGKs that were delivered to the device. The SGKs used in this process belong to the same RI context as the BCRO.

- Only members of the subscriber group can use the content key. If a member of the group succeeds in constructing the IEK, that member can decrypt the content key. Any group member that tries to deduce the IEK but does not have the appropriate (zero message) SGK on board is unable to deduce the IEK to decrypt the content key.

5.5.2.4 Addressing a unique device

To access a unique device, the following (oversimplified) BCRO is used (see Figure 14).

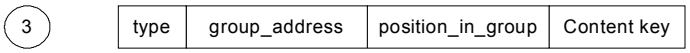


Figure 14 – (Oversimplified) unique device BCRO

A unique device is a privileged set of the total group.

- A unique device is addressed by a unique address, which is a group address plus a position/offset in the group. This address matches the unique_device_filter address that was sent with the registration data to the device (see 9.3.2.7 for details).
- The content key (which can carry the SEK and/or PEK) is encrypted with an inferred encryption key (IEK). In this case, the unique device key (UDK) is used to derive the IEK, see 5.4.7. The UDK used by the RI is identical to the key that was delivered with the registration data sent to the device (see 9.3.2.7 for details).
- Only the unique device that is addressed can use the content key.

In all cases, the head end infrastructure composes the IEK used to encrypt the content key and determines the access mask on the basis of the created key. For the subscriber group case the head end infrastructure creates the access mask based on the corresponding (zero message) SGKs. A "type" field inside the BCRO shall indicate which addressing case is covered.

5.5.2.5 Addressing a broadcast domain

To access a broadcast domain, the following (oversimplified) BCRO is used (see Figure 15).

Figure 15 – (Oversimplified) broadcast domain BCRO

A broadcast domain is a privileged set of the total group.

- The domain address was delivered with the registration data (see 9.3.2.7 for details) and/or via a join domain response (see 9.3.3.5.1.1 for details) in the form of the shortform_domain_id and is used to filter for the message. This address is the OMA DRM 2.0 domain ID.
- The content key (which can carry the SEK and/or PEK) is encrypted with an inferred encryption key (IEK). In this case, the broadcast_domain_key (BDK) is used to derive the IEK, see 5.4.7. The BDK used by the RI is identical to the key that was delivered with the registration data sent to the device (see 9.3.2.7 for details).
- All the devices in a broadcast domain group can use the content key.

NOTE Broadcast domain addressing is included in this standard as an additional addressing option that will potentially save some bandwidth over unique device addressing, because more devices belonging to one user might be registered into the same broadcast domain group. The "savings" in bandwidth with domain addressing are not as high as under subscriber group addressing. For completeness, it is mentioned that the domain addressing can also be used in another mode: an option would be to use domain addressing with a population of millions of devices to allow large groups to access low value content. In this particular use of domains, the "savings" in bandwidth might be considerable compared to any other of the mentioned addressing modes. The trade-off is that a security incident can affect more devices.

5.5.3 Zero message broadcast encryption scheme

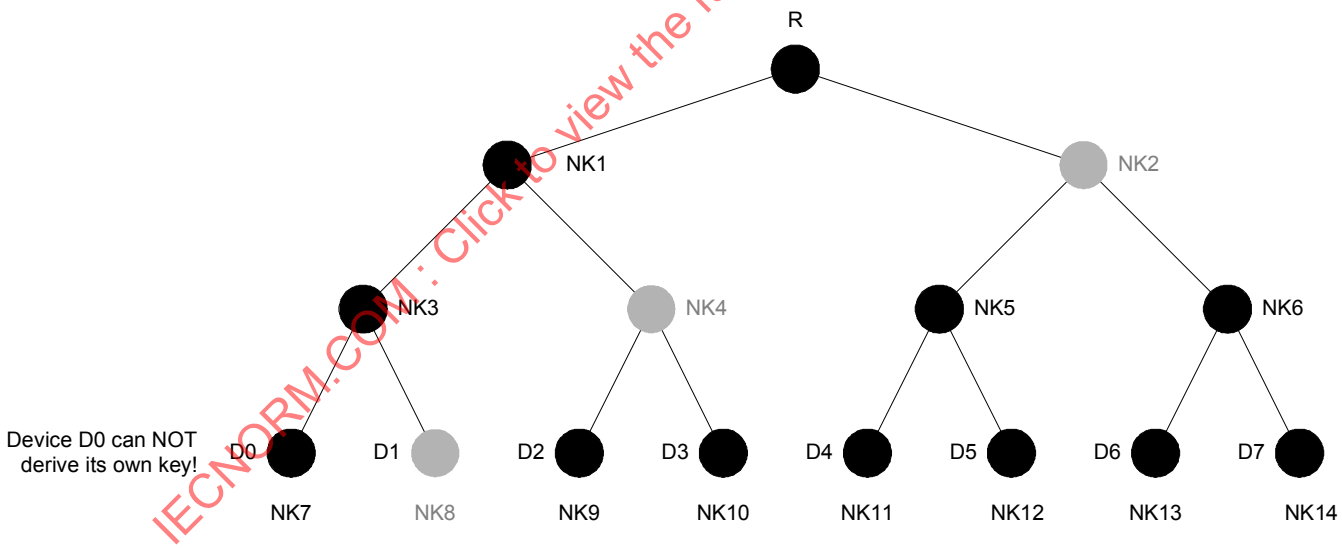
Zero message broadcast encryption was originally introduced in Fiat Naor:1998. It solves the problem of creating a privileged set within a group of devices without the need for sending out messages to create such a set. During device registration, the RI creates registration data for the device and is able to enter the correct subscriber_group_keys (SGK) into the registration data. After the device registration, no more data needs to be transmitted to the unique Devices, which is why it is called a zero message broadcast (a.k.a. ZMB) scheme. This, therefore, preserves bandwidth on the network.

This standard uses an algorithm similar to the Fiat-Naor scheme, but with the following advantages.

- The algorithm is altered to support changing the IEK independently of the subscriber group keys.
- The IEK does not reveal information about the subscriber group keys.

Zero message broadcast encryption is based on a set of group keys that are provisioned to the terminal during registration. The number of group keys needed depends on the group size (n = log2(group size)). Each terminal needs (worst case) to derive m -1 keys (with m = group size). A device that implements zero message broadcast encryption will need, for group addressing, n + 1 keys (the additional key is used when the privileged set is equal to the complete group).

Deriving the required keys is done as follows, illustrated with a group size of 8 in Figure 16.



NOTE Key material for the authentication of messages is omitted for readability reasons.

Figure 16 – Example of a zero message tree with three nodes (keys)

m = 8, the number of terminal keys will be log2(8) = 3

Assume the terminal D0 has the following keys:

{NK2, NK4, NK8}

The terminal now derives, as specified in Clause A.7 steps a to d, a set of leaf node keys {D4, D5, D6, D7} from NK2 and {D2, D3} from NK4. This is done using AES such that the left child key of key NKi is AES_ENC_128{NKi}(T1+2i) and the right child key is AES_ENC_128{NKi}(T2+2i), see also A.7.1. Using the mechanisms as set forth in Clause A.7, the device can derive the leaves in the following order.

- From NK2 it derives NK5 and NK6.
  - From NK5 it derives NK11 and NK12
  - From NK6 it derives NK13 and NK14
- From NK4 it derives NK9 and NK10

The || binary operator is used to denote concatenation. The notation HMAC_SHA1{k}(s) is used to denote the computation of HMAC, see IETF RFC 2104, keyed by the key 'k' over the string 's' with SHA1, see FIPS PUB 180-2:2002, as the hash function. HMAC_SHA1_128{k}(s) is used to denote the 128 most significant bits of HMAC_SHA1{k}(s) output. The IEK is constructed by computing HMAC_SHA1 keyed by the concatenation of the selected keys, with the keys ordered according to their index in the Eurocrypt address (see EN 50094:1992). The HMAC_SHA1 is computed over a salt that is defined as follows. The IEK is specific to each encrypted SEAK or PEAK in the BCRO. The 96-bit BCI field from the BCRO asset() structure is used as the salt. The BCI value from the first asset structure in a BCRO shall be used for all assets in a BCRO structure in the case that there is more than one asset structure in a BCRO.

<salt> = BCI

IEK = HMAC_SHA1_128{NK8 || NK9 || NK10 || NK11 || NK12 || NK13 || NK14}(<salt>).

NOTE 1 To exclude a device from the so-called privileged set, its key is included in the decryption key. If terminal D1 and D5 are to be excluded, the rights decryption key would be constructed by computing HMAC_SHA1_128{D1 || D5}(<salt>). Given that terminal D1 and D5 are not able to derive their own keys (i.e. D1 or D5), these terminals are excluded from the so-called privileged set and cannot create the rights decryption key. See Clause A.7 for an overview of the function F_{ZMB}.

NOTE 2 The best (feasible) group sizes will be 256 or 512. Because of the small size of the group, the local processing requirements are limited on the device side, and the device needs limited resources for the Fiat Naor tree. The derivation process will have some computational overhead but can be implemented efficiently (for example, it is not necessary to calculate all terminal keys before processing them into the broadcast key).

One of the reasons to keep the group size small is because the security of the zero message broadcast encryption scheme is based on the assumption that any two devices within the same group do not share or exchange their broadcast encryption keys. This means that if two members of a group hack their terminal and merge their subscriber group key set, they would be capable of a collusion attack which would enable them to decrypt any messages that are addressed to a subset of this group – even the ones for which they are not authorized. The RI would be forced to re-register the complete group (distributed over a number of groups to perform some sort of a stepwise refinement scheme to identify the culprits). However, when the group sizes are small enough (i.e. 256 or 512), collusion is not a big threat.

This scheme improves upon the current practice of relying purely on tamper-resistance, as breaking the tamper-resistance of a single device is not sufficient to obtain access to unauthorized content. A collusion attack where the SGK keys from at least two members of the same group is required. Even when the keys have been acquired, these keys need to be distributed to another device, which is by no means trivial. Above that a distribution might be also be traceable.

In order to make the probability of such collusion attacks negligible, the following criteria should be followed when assigning devices to broadcast encryption groups.

a) Devices owned by the same user are not assigned to the same group.

b) Each device is assigned to a group randomly.

It is the responsibility of the RI to make available and manage an appropriate number of groups. When the number of groups is in the order of thousands, collusion attacks become costly and impractical.

6 Traffic layer

6.1 General

In this clause, the traffic layer is specified. The traffic layer is layer 1 of the four-layer model (see 5.4) used in the protection functionality specified by this standard. The specific use of the traffic layer for the individual supported systems is further specified in 13.2, 14.2, 15.3 and 16.2.

For protection of media streams on the traffic layer, either IPsec, SRTP, ISMACryp or MPEG2 TS CRYPT shall be used, as defined in detail in this clause. Furthermore, this clause explains how to use these protection methods together with the other layers defined in this standard. Specific normative prescriptions for the cipher choice apply for each of the covered broadcast applications, as specified in Clauses 13, 14, 15 and 16.

6.2 IPsec

6.2.1 General

The broadcast network may use IPsec, see IETF RFC 4301, to protect broadcast services; see Clauses 13 and 16.

The IPsec implementation in the device shall be such that it does not interfere with the usage of IPsec for other applications. This implies that the security parameter index (SPI) allocation and security association (SA) look-ups shall be implemented in such a way that they interoperate with existing IPsec implementations.

An IPsec SA consists of a tuple of the following parameters:

- selectors (IP protocol version, source IP address, destination IP address, protocol, source port and destination port);
- SPI;
- destination IP address;
- security protocol, security protocol mode and security protocol parameters;
- algorithms and algorithm parameters;
- key material.

An IPsec SA shall be uniquely identified by a destination IP address and SPI pair.

Figure 17 shows the different objects and elements involved in instantiating IPsec security associations.

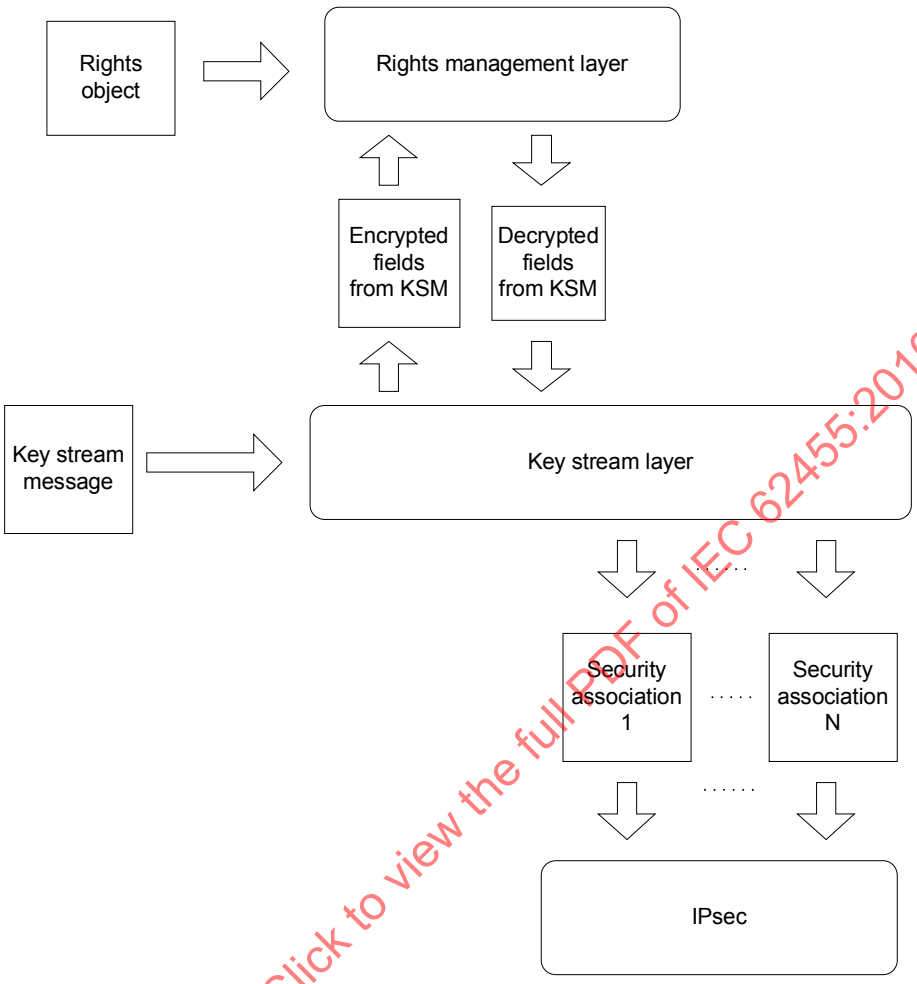


Figure 17 – IPsec security association elements

The instantiation of security associations is performed by the key stream layer and is driven by key stream messages and rights objects. When a key stream message is received, the key stream layer extracts the encrypted fields from that message. The key stream passes these and other relevant fields to the rights management layer. For each rights object stored in the device, the rights management layer determines whether that rights object would be able to decrypt the fields in the key stream message. If the rights layer finds a suitable rights object, it decrypts these fields using the appropriate rights management system and rights object. The decrypted fields are provided back to the key stream layer, which – based on the key stream message and the decrypted fields – instantiates a set of security associations. If the rights management layer does not find a suitable rights object, the key stream message should be silently dropped.

6.2.2 Selectors

Selectors are provided by the KSL. The selectors may contain wildcards, ranges or point values, but all the other parameters shall be exactly defined. All address selectors shall be point values and the destination address selector shall match the destination IP address of the SA.

6.2.3 Encapsulation protocol and mode

If IPsec is used for traffic encryption, the protocol and mode shall be ESP in transport mode, according to IETF RFC 4301 and IETF RFC 2406. Other IPsec encapsulation protocols or modes shall not be used.

6.2.4 Encryption algorithm

The encryption algorithm for IPsec ESP shall be AES-128-CBC with explicit IV in each IP packet, as defined in IETF RFC 2451 and IETF RFC 3602. Other encryption algorithms, key sizes or chaining modes shall not be used.

6.2.5 Authentication algorithm

The authentication algorithm for IPsec ESP shall be HMAC-SHA-1-96, as defined in IETF RFC 2104 and IETF RFC 2404. Other authentication algorithms or truncations shall not be used.

The system may use authentication. If no authentication is desired, the NULL authentication algorithm shall be specified. In this case, replay protection shall not be performed by the device.

6.2.6 Security association management

The KSL defines how often the transport layer keys are re-keyed. This sets the following requirements.

- The TEK provided by KSL shall be used as the key for the ESP encryption.
- The traffic authentication key (TAK) provided by the KSL shall be used as the key for the ESP message authentication code (MAC) if authentication is used.
- The IPsec implementation shall be able to manage SAs relating to the KSL separately from those managed manually or by any other protocol such as IKE. This implies the ability to identify whether an SA relates to the KSL.
- SAs relating to KSL shall be prioritised lower than those SAs that have a locally defined policy or a policy that is provided by a trustworthy party.
- SAs relating to KSL are simplex and shall be applied only to inbound traffic on the recipient side.

The re-keying of existing SAs by the KSL should be managed on a resource basis by the IPsec layer according to the following recommendations.

- The IPsec implementation should be able to keep alive at least the two most recently instantiated IPsec SAs for a particular set of selectors.
- The IPsec implementation should provide a least-recently-instantiated mechanism for cleaning up SAs as resources reserved for IPsec SAs are exhausted.
- The number of IPsec SAs required to exhaust the resources such that the clean-up mechanism is triggered should be 3 per service key per set of IP selectors.
- A device should be able to re-key any SA at least for every 20 received ESP packets without a significant loss in performance. This re-key consists of installing a new SA with a defined set of selectors, and possibly, eliminating an old SA with an equal set of selectors. Both SAs in this case are managed by the KSL.

  A broadcaster is not recommended to re-key existing SAs for every 20 packets, as the amount of traffic one can place in 20 packets varies heavily with the maximum packet size. The impact on the device in terms of time is also hard to estimate, as the timing between packets may be significantly altered in a broadcasting environment. Therefore, a broadcaster shall not re-key an IPsec SA more often than every 2 s at the point of sending the messages through KSL.

6.3 ISMACryp

6.3.1 Streamed content

Streamed content may be encrypted and carried over RTP as specified in ISMA Encryption and Authentication (refer to ISMACryp 2.0); see Clauses 13 and 16.

The ISMACrypSalt parameter shall be used to signal the ISMACryp salt in the attributes of each encrypted media streams.

The default ISMACryp cipher, mode and configuration, AES-128-CTR, as defined in ISMACryp 2.0, shall be used.

An ISMACryp broadcast implementation shall change the key over time, but no faster than once per second for each encrypted stream.

The length of the key indicator shall be greater than or equal to 1 byte.

6.3.2 Downloadable audio/visual content (stored in MP4 files)

This subclause applies to the use of a broadcast system for the distribution (download) of protected files containing, for example, audio/visual content, instead of to the transmission of a broadcast stream. The difference between streaming and file download is explained below.

- A broadcast stream can be rendered by a device irrespective of the point in the stream from which the device starts receiving it.
- A device can only start to render a file if it has received at least the first part of that file (progressive download) or potentially even only after it has received that entire file (download). How much of the file needs to be received before the device can start rendering depends on the distribution of information within the file.

Encryption of downloadable audio/visual content, when stored in an ISO base media file format file (MP4 file, see ISO/IEC 14496-12:2005), may be performed as specified in ISMA Encryption and Authentication, see ISMACryp 2.0.

The default ISMACryp cipher, mode and configuration, AES-128-CTR, as defined in ISMACryp 2.0, shall be used.

The ISMACrypSaltBox box shall be used to signal the ISMACryp salt in the SchemeInformationBox of each encrypted media stream.

An ISMACryp broadcast implementation shall change the key over time, but no faster than once per second for each encrypted stream.

To support a SimulCrypt approach, see ETSI TS 103 197, the SchemeInformationBox of each encrypted media stream may contain multiple occurrences of the ISMAKMSBox, one for each KMS.

6.3.3 Use of ISMACryp with the rights management and key stream layers

Figure 18 shows how ISMACryp is used with the rights management and key stream layers.

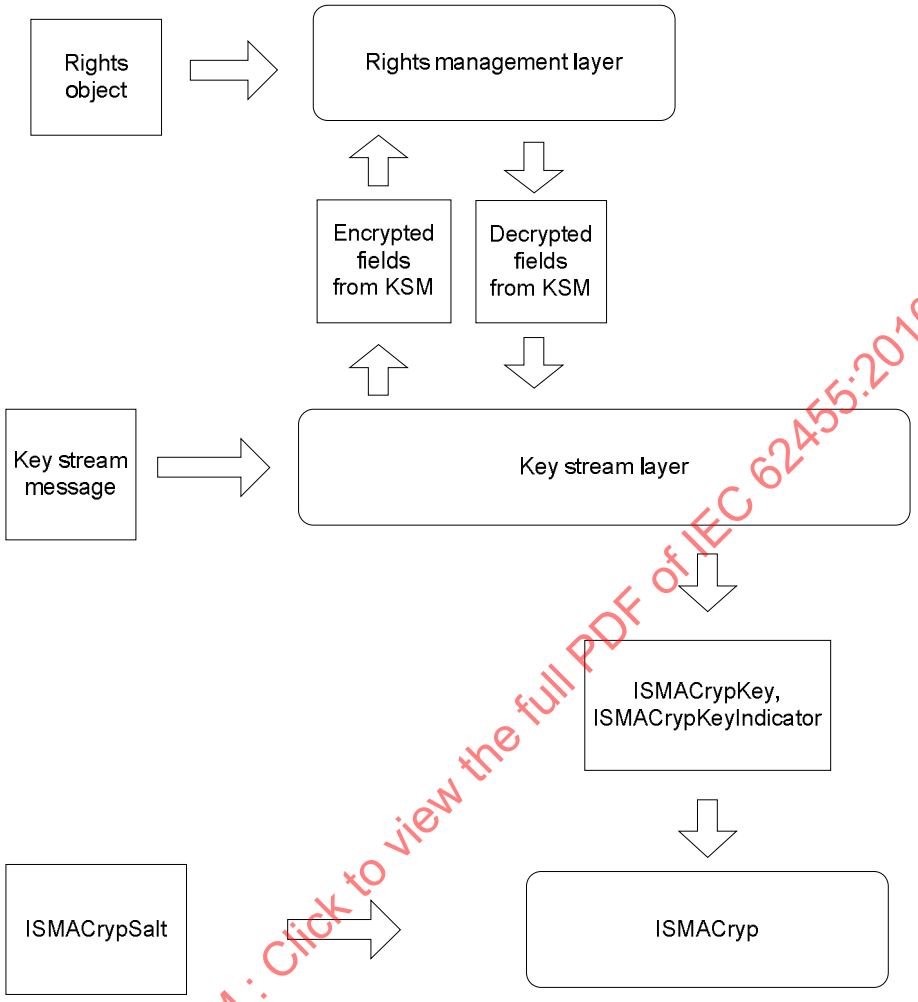


Figure 18 – ISMACryp Key Management

When ISMACryp is used, the key stream message carries encrypted ISMACrypKey in the traffic key material field, and ISMACrypKeyIndicator, which is used to associate the ISMACrypKey with a particular access unit of the content stream. Given a key stream message, the key stream layer extracts the encrypted fields from that message. The encrypted fields are sent to the rights management layer for decryption. If the rights management layer finds a suitable rights object, it decrypts the fields using the appropriate rights management system and rights object. The decrypted ISMACrypKey is sent along with the ISMACrypKeyIndicator to the ISMACryp content decrypter. When an access unit with a matching ISMACrypKeyIndicator is received, the ISMACryp decrypter uses the ISMACrypKey to decrypt the content. The ISMACrypSalt parameter is signalled in the SDP attributes; see IETF RFC 2327 of the stream. If the rights management layer does not find a suitable rights object, then the key stream message should be silently dropped.

6.4 SRTP

6.4.1 General

The broadcast network may use SRTP, see IETF RFC 3711, to protect broadcast services; see Clauses 13 and 16.

An SRTP session is defined as the cryptographic context for an RTP session. The cryptographic context for SRTP, when used for service protection, consists of the following elements:

- roll-over counter (ROC);
- receiving sequence number;
- cipher and mode definition;
- message authentication code (MAC) method definition;
- list of received packets;
- array of master keys (MKs);
- master key indicator (MKI) indicator bit;
- length of the MKI field;
- value of currently active MKI;
- array of counters of processed packets for each master key;
- length of encryption and authentication keys;
- master salt;
- context id.

A cryptographic context is uniquely identified by its context id. The context id consists of the synchronization sources, destination network address and destination transport port number.

Figure 19 shows the different objects and elements involved in building SRTP cryptographic context.

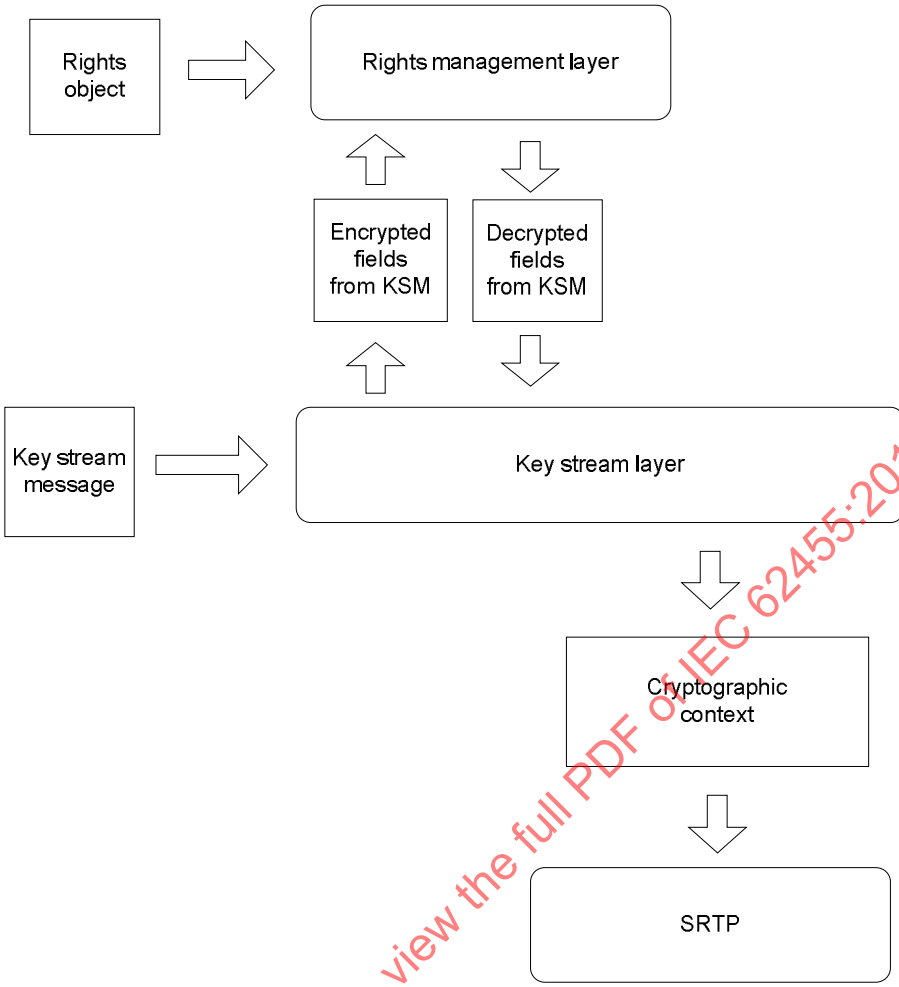


Figure 19 – SRTP cryptographic context management

As with IPsec security associations, the instantiation of a cryptographic context is performed by the key stream layer and is driven by key stream messages and rights objects. A key stream message includes parameters that are specific to the selected content encryption layer. In the case of SRTP, a key stream message includes an MKI (master key index) that is necessary to identify an SRTP cryptographic context. The SRTP ROC (roll-over-counter) values for each component stream are communicated in the integrity transform of SRTP packets in order to keep track of how many times each RTP sequence number has wrapped around. This is required for keeping receivers synchronized with the current SRTP session. Because it is possible that a received ROC value is already out of date by the time that a receiver starts decrypting packets, there is additional information provided in the SRTP packets that allows receivers to adjust the ROC value appropriately, based on the specified algorithm.

Just as it is the case with IPsec, an encrypted traffic key is extracted from a key stream message and is passed to the rights management layer for decryption. If the rights management layer finds a valid rights object for this traffic key, it will be decrypted and converted to an SRTP master key. The SRTP content decryption layer then creates an SRTP session with decryption and (optionally) authentication keys that are derived from the master key as required by SRTP. If the rights management layer does not find a suitable rights object, then the key stream message should be silently dropped.

6.4.2 Key management

The SRTP application shall use the MKI value for looking up decryption keys. This means that a cryptographic context shall have the MKI indicator bit set to 1. The <From, To> value method of key lookup shall not be used.

The master salt shall not be used.

The TEKs provided by the KSL shall be used as the SRTP MK.

The key derivation rate shall be 0. Exactly one SRTP session encryption key shall be derived from one MK. If SRTP authentication is enabled, exactly one SRTP session authentication key shall be derived from one MK.

The KSL shall provide and update the cryptographic contexts to the SRTP implementation (excluding the ROC). Some fields are initialized and/or managed internally within the SRTP implementation, such as the list of received packets used in replay protection, receiving sequence number, and the ROC.

The ROC value is included in the plaintext over which a MAC is computed (assuming authentication is used). The ROC value is included in the (implicit) IV for the AES-CTR encryption, and therefore the ROC is needed to encrypt/decrypt a packet.

The Sender's ROC shall be transferred in every R-th packet according to IETF RFC 4771, where R is a configurable parameter that is signalled out of band and shall be greater than 0. See 13.6.2 for the out-of-band signalling.

Because the SRTP key-derivation rate is not used and the <From, To> values are also not used, the SRTP crypto context will be re-keyed by the KSL. The SRTP implementation shall be able to handle installing a new crypto context every 20 packets. An SRTP broadcasting implementation shall not require an install or an update of a new crypto context more than once a second for a single SRTP context id, at the point of sending the messages through the KSL.

6.4.3 Encryption algorithm

The encryption algorithm for SRTP packets shall be AES-128-CTR, as defined in IETF RFC 3711. Other encryption algorithms, key sizes or chaining modes shall not be used.

6.4.4 Authentication algorithm

The authentication algorithm for SRTP shall be as defined in IETF RFC 4771, based on HMAC-SHA-1-80 as defined in IETF RFC 2104 and IETF RFC 3711. Other authentication algorithms or truncations shall not be used.

IETF RFC 4771 defines two versions of a roll over counter carrying (RCC) transform, both of which require authentication of those packets containing the sender's ROC. The broadcast system may use authentication on the remaining packets. Those packets for which no authentication is desired shall not contain an SRTP authentication tag field (see IETF RFC 4771, Clause 3). In this case, also replay protection shall not be performed by the device.

The version of RCC transform used is a configurable parameter that is signalled out of band. See 13.6.2 for the out-of-band signalling.

6.5 MPEG2 TS crypt

6.5.1 General

Figure 20 shows the different objects and elements involved in building MPEG2 transport stream cryptographic context for MPEG2 TS CRYPT.

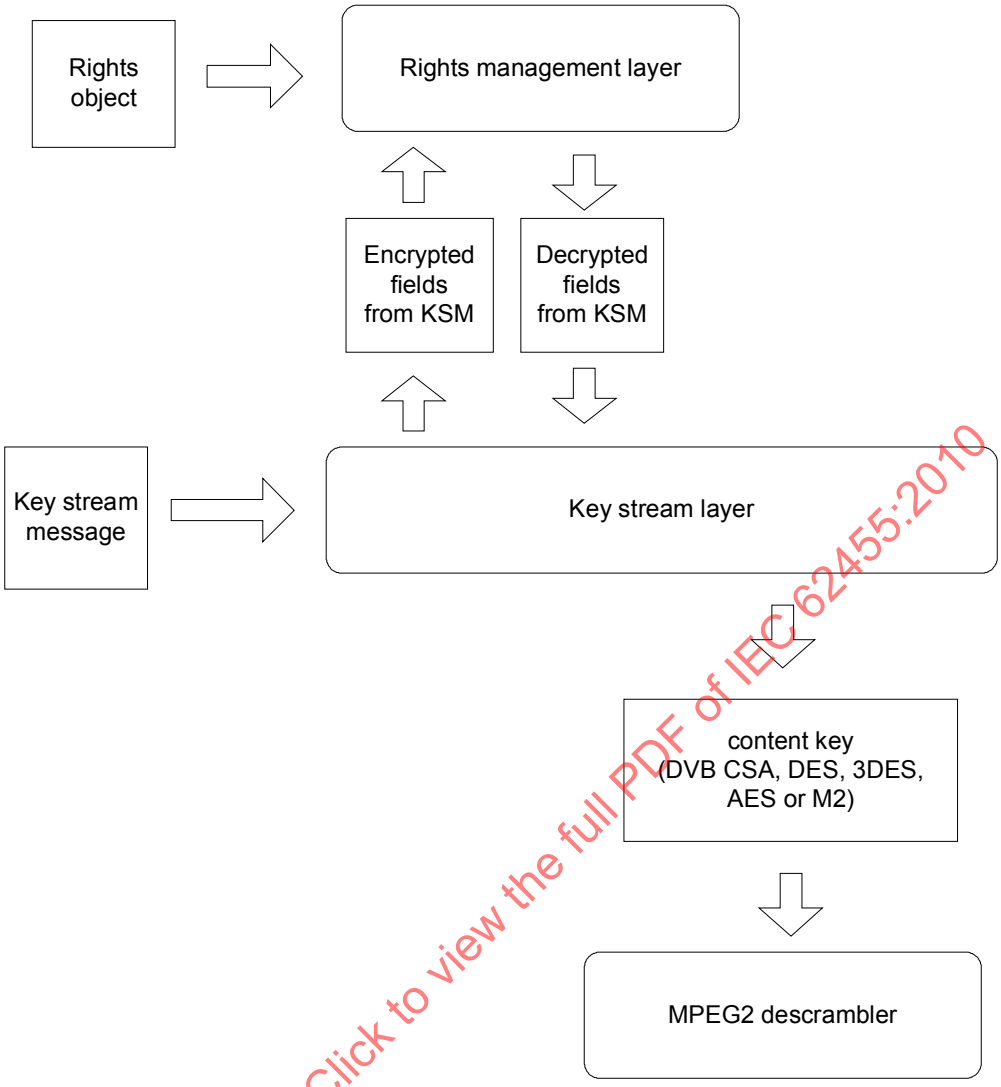


Figure 20 – MPEG2 transport stream cryptographic context management

The decryption of a content key shall be performed by the key stream layer and shall be driven by key stream messages and rights objects. A key stream message includes parameters that are specific to the selected content protection method in the traffic layer. In the case of MPEG2 TS CRYPT protection as specified in this standard, a key stream message shall include a content key, which is necessary in an MPEG2 cryptographic context.

The following analogy is given to clarify the vocabulary used in this standard with the typical vocabulary used in an MPEG2 context with DVB-CSA.

In an MPEG2 environment with DVB-CSA, there are three different credentials involved in the (de)scrambling.

a) An ECM carries a "control word", which is used to initialize the descrambling.

b) An EMM carries a "service key", which is used to encrypt the control word for a group of one or more users.

c) A "user key", contained in for instance a smart card, which is used to encrypt the service key.

In an MPEG2 environment with DVB-CSA the decryption in a device consists of recovering the service key from the EMM and the user key. The service key is then used to decrypt the ECM in order to recover the control word. Loading the "control word" into the odd and/or even registers starts the initialization process of the descrambler.

In this standard, descrambling of scrambled AV content carried over a MPEG2 TS is supported in the following way. The decryption in a device consists of recovering the "service" (or programme) key from the RO and the "user key" from the registration data. The service or programme key is then used to decrypt the KSM in order to recover the "control word" allowing the initialization of the descrambler.

a) The "user key" is carried by the registration data, which is protected by the device private key.

b) The RO carries the "service" (or programme) key.

c) The KSM carries the key material, being in this DVB-CSA analogy the "control word".

The "control word" for the descrambler is called content key in this standard. The content key can be a DVB-CSA key or a DES, 3DES, AES or M2 key.

The scrambling method used for AV content carried in an MPEG2 TS shall be either transport stream level scrambling or PES level scrambling.

6.5.2 Transport stream level scrambling

To protect network broadcasts the broadcaster may "scramble" (a.k.a. encrypt) the transport stream (a.k.a. TS). In this case, scrambling takes place after multiplexing the payload of the transport packet. The receiving device will have to "descramble" (a.k.a. decrypt) the TS so the audio and/or video and/or data parts can be consumed. This is done with a piece of hardware called the "descrambler". The descrambler knows when it has to descramble or not by looking at the transport stream control (a.k.a. TSC) bits in the TS packet as defined in Table 3.

Table 3 – Definition of transport_scrambling_control bits

Transport stream control bits	Description
00	No descrambling
01	Scrambling with the DEFAULT content key
10	Scrambling by the EVEN content key
11	Scrambling by the ODD content key

Limitations to TS level scrambling will adhere to ISO/IEC 13818-1.

6.5.3 PES level scrambling

Instead of scrambling all the content at the TS level, one or more of the packetized elementary streams (a.k.a. PES) may be scrambled. In this case, scrambling generally takes place at the source, before multiplexing. The descrambler knows whether to descramble or not by looking at the 2-bit PES scrambling control field in the PES packet header as defined in Table 4.

Table 4 – Definition of pes_scrambling_control field bits

PES_scrambling control field	Description
00	No descrambling
01	No descrambling
10	Scrambling by the EVEN content key
11	Scrambling by the ODD content key

Limitations to PES level scrambling will adhere to ISO/IEC 13818-1.

6.5.4 Descrambling MPEG2 content

An encrypted traffic key is extracted from a key stream message and is passed to the rights management layer for decryption. If the rights management layer finds a valid rights object for this traffic key, it will be decrypted and converted to a content key. If the rights management layer does not find a suitable rights object, then the key stream message should be silently dropped.

Independent of TS or PES level descrambling, the descrambler shall use content keys for descrambling scrambled (TS or PES) packets.

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

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

There are two possible descrambler implementations: single-key and dual-key descramblers; see also Figure 21.

- Single-key descramblers have one register to store a descrambler key. To change the key they have to overwrite the key in the register. This cannot be done when the descrambler is in the middle of descrambling a packet. Therefore, the broadcaster usually alternates the scrambled transmission by transmitting packets in the clear a short interval (mostly 1 s). The descrambler's register is then updated with the new key. After the clear interval expires, the broadcaster sends TS packet that are scrambled with the new key. This cycle can repeat itself endlessly.
- Dual-key descramblers have two registers so they can store two keys: the first register can contain the key the descrambler is currently using and the second register can be updated with a new key for the next keying period. To distinguish the registers they are identified as the odd and even key register. The TSC bit in the TS packet indicates if the descrambler needs to use the key in the odd or even key register in order to descramble the TS packet and flips to corresponding register when necessary.

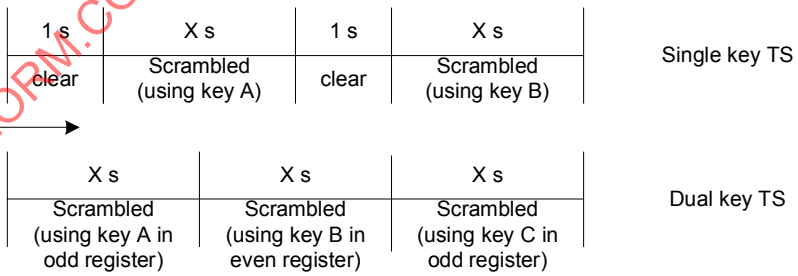


Figure 21 – Single-key versus dual-key TS over time

The KSM signals if the delivered traffic key is "odd" or "even". Because of the aforementioned behaviour, Table 5 applies.

Table 5 – Descrambling possibility matrix

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

[a] TSC bits in single key TS will be stationary odd or even. The following two solutions are possible:

a) detect TSC status and put key in correct register; or

b) put the existing key in both the odd and even register at the same time.

[b] There is no second in the clear in dual-key TS to change the key.

6.5.5 Supported ciphers

For MPEG2 TS Crypt, this standard supports the ciphers listed in Table 6.

Table 6 – Supported ciphers for MPEG2 TS Crypt

Cipher	Mandatory / optional	Reference
DVB-CSA with 64-bit key	Mandatory[a]	ETSI ETR 289
DES	Optional [a]	FIPS PUB 46-3:1999, FIPS PUB 81:1980
3DES-168	Optional [a]	FIPS PUB 46-3:1999, FIPS PUB 81:1980
3DES-112	Optional [a]	FIPS PUB 46-3:1999, FIPS PUB 81:1980
3DES-56	Optional [a]	FIPS PUB 46-3:1999, FIPS PUB 81:1980
AES with 128-bit key	Mandatory [a]	FIPS PUB 197:2001
M2 with 64-bit key	Optional [a]	

[a] A Device implementing "MPEG2 TS Crypt"

- shall implement the DVB-CSA as well as the AES-128 cipher; the other ciphers as indicated in Table 6 may be implemented,
- shall support the interpretation all values of the content_key_index field in the KSM (see 7.2).

NOTE 1 It is out of the scope of this standard how the RI would get knowledge of the device capabilities. Such issues may be sorted out between the RI and ROT based on the UDN.

NOTE 2 There are differences in bit and byte numbering used in MPEG (see ISO/IEC 13818-1) on the one hand and the one used in the specification of DES and 3DES (see FIPS PUB 46-3:1999) on the other hand. How to map the one numbering system to the other is defined in ATSC Doc. A/70A:2004, Annex A.

6.5.6 Key management

Under MPEG2 TS crypt, several ciphers can be used, such as DVB-CSA, DES, 3DES, AES or M2. Irrespective of the cipher used the odd_even_flag shall be in line with the bits of the transport_scrambling_control bits (see 6.5.2) or the pes_scrambling_control bits (see 6.5.3) Using Table 5, the device will decide how to handle encrypted_traffic_key_material and next_encrypted_traffic_key_material in odd and/or even registers.

The content_key_index field in the key stream message indicates which cipher is used. The list below specifies some cipher dependent functionality.

- All ciphers

  The odd_even_flag in the key stream message will indicate if the encrypted_traffic_key_material field will contain a content key for the odd or even register. The use of odd and even is explained in 6.5.2 and 6.5.3. Shortly before the traffic key lifetime expires, the KSM will include the field next_encrypted_traffic_key_material to allow the device to prepare for the key change.

NOTE Although odd/even functionality is in practice not used for other ciphers than DVB-CSA, this standard supports this functionality for all ciphers.

- DES, 3DES, AES

  DES, 3DES and AES can be used in ECB or CBC mode. In ECB mode, there is no IV. In CBC mode, the KSM shall include an IV. Switching from one key and IV to another is done when the traffic key lifetime expires. Shortly before the traffic key lifetime expires, the KSM will include the field next_encrypted_traffic_key_material and the field next_initialisation_vector to allow the device to prepare for the key change. The termination block handling is specified in ANSI/SCTE 52:2008 and shall be applied when the last content block to be encrypted is smaller than the cipher block size.

7 Key stream layer

7.1 General

In this clause, the key stream layer is specified. The key stream layer is layer 2 of the four-layer model (see 5.4) used in the protection functionality specified by this standard. The specific use of the key stream layer for the individual supported systems is further specified in 13.3, 14.3, 15.4 and 16.3.

The key stream layer is made up of key stream messages, which shall be distributed in-band, together with the protected media streams.

7.2 Format of the key stream message (KSM)

7.2.1 Format

The format of the key stream message is specified in Table 7.

Table 7 – Format of key stream message

Field	Length	Type
key_stream_message() {		
selectors_and_flags {		
protocol_version	4	uimsbf
reserved_for_future_use	3	bslbf
access_criteria_flag	1	uimsbf
traffic_protection_protocol	3	uimsbf
traffic_authentication_flag	1	uimsbf
next_traffic_key_flag	1	uimsbf
timestamp_flag	1	uimsbf
programme_flag	1	uimsbf
service_flag	1	uimsbf
}		
if (traffic_protection_protocol == KSM_ALGO_IPSEC) {		
security_parameter_index	32	uimsbf
if (next_traffic_key_flag == KSM_FLAG_TRUE) {		
next_security_parameter_index	32	uimsbf
}		
}		
if (traffic_protection_protocol == KSM_ALGO_SRTP) {		
master_key_index_length	8	uimsbf
master_key_index	8*master_key_index_length	uimsbf
reserved_for_future_use	7	bslbf
master_salt_flag	1	uimsbf
}		
if (traffic_protection_protocol == KSM_ALGO_ISMACRYP){		
key_indicator_length	8	uimsbf
key_indicator	8*key_indicator_length	bslbf
if (next_traffic_key_flag == KSM_FLAG_TRUE) {		
key_indicator	8*key_indicator_length	bslbf
}		
}		
if (traffic_protection_protocol == KSM_ALGO_MPEG2_TS_CRYPT) {		
content_key_index	4	uimsbf
odd_even_flag	1	bslbf
cipher mode	3	uimsbf
reserved_for_future_use	8	bslbf
if (cipher_mode == 0x1) {		
initial_vector_length	8	uimsbf
initial_vector	8 * initial_vector_length	bslbf
if (next_traffic_key_flag == KSM_FLAG_TRUE){		
next_initial_vector	8 * initial_vector_length	bslbf
}		

Field	Length	Type
}		
}		
encrypted_traffic_key_material_length	8	uimsbf
encrypted_traffic_key_material	8*encrypted_traffic_key_material_length	bslbf
if (next_traffic_key_flag == KSM_FLAG_TRUE) {		
next_encrypted_traffic_key_material	8*encrypted_traffic_key_material_length	bslbf
}		
reserved_for_future_use	5	bslbf
traffic_key_lifetime	3	uimsbf
if (timestamp_flag == KSM_FLAG_TRUE) {		
timestamp	40	mjdutc
}		
if (access_criteria_flag == KSM_FLAG_TRUE) {		
reserved_for_future_use	8	bslbf
number_of_access_criteria_descriptors	8	uimsbf
access_criteria_descriptor_loop() {		
access_criteria_descriptor()		
}		
}		
if (programme_flag == KSM_FLAG_TRUE) {		
programme_selectors_and_flags {		
reserved_for_future_use	7	bslbf
permissions_flag	1	uimsbf
}		
if (permissions_flag == KSM_FLAG_TRUE) {		
permissions_category	8	uimsbf
}		
if (service_flag == KSM_FLAG_TRUE) {		
encrypted_PEK	128	bslbf
}		
programme_CID_extension	32	uimsbf
programme_MAC	96	bslbf
}		
if (service_flag == KSM_FLAG_TRUE) {		
service_CID_extension	32	uimsbf
service_MAC	96	bslbf
}		
}		

7.2.2 Descriptors for access_criteria_descriptor_loop

7.2.2.1 General

The format of the access_criteria_descriptor is specified in Table 8.

Table 8 – Descriptors for access_criteria_descriptor_loop

Field	Length	Type
tag	8	uimsbf
length	8	uimsbf
value	8*length	bslbf

The access criteria descriptor loop is an extension mechanism to allow the addition of new access criteria in the future versions of this standard. The device shall ignore access criteria descriptors that it does not support.

A single access criteria descriptor can carry one or more access criteria.

The access criteria descriptors listed in Table 9 have been defined. They are specified in the following subclauses.

Table 9 – Access_criteria_descriptors

Access_criteria_descriptor	Tag	Comment
parental_rating	0x01	
copy_control_information	0xE1	CCI information byte; not applicable to IPDC over DVB-H systems
blackout_spotbeam	0xE2	Not applicable to IPDC over DVB-H systems

7.2.2.2 Parental_rating access criteria descriptor

This descriptor represents the parental rating of the programme. The descriptor tag for this descriptor is defined in Table 9. This descriptor is encoded as specified in Table 10.

Table 10 – Parental_rating access criteria descriptor

Field	Length	Type
rating_type	7	uimsbf
country_code_flag	1	uimsbf
rating_value	8	uimsbf
if (country_code_flag == KSM_FLAG_TRUE) {		
number_of_country_codes	8	uimsbf
for (i = 0; i < number_of_country_codes; i++) {		
country_code	16	uimsbf
}		
}		

The optional list of country_code(s) specifies that the rating is for a specific list of one or more countries, which is analogous to the MPEG-7, see ISO/IEC 15938-5:2003, definition of the

ParentalGuidanceType. Each country code is a 2-character value that shall be compliant with ISO 3166.

The rating_type with values 0 through 8 specifies one of the content rating systems that are defined by MPEG-7, see ISO/IEC 15938-5:2003, and the rating value is an integer with the meaning that is dependent on the rating_type; see Table 11. The rating values for rating type 0 through 8 are exactly as they had been defined by MPEG-7; see ISO/IEC 15938-5:2003. Rating type 9 is for the parental rating for the German system; see Table 11.

Table 11 – Parental rating values for each parental rating type

Rating _type	Name	Description	Rating_value
0	N/A	ETSI EN 300 468 parental_rating_descriptor in DVB systems	Minimum allowable age, encoded as specified in ETSI EN 300 468 for the field 'rating' of parental_rating_descriptor
1	JapaneseAdmCommMotionPictureCodeEthicsParentalRatingCS	Japanese Motion Picture Parental Rating	0 = G 1=PG12 2=R-15 3=R-18 4=None
2	ICRAParentalRatingCS	Internet Content Rating Association Parental Rating	1=Level4 2=Level3 3=Level2 4= Level1 5= Level0 6= None
3	MPAAParentalRatingCS	MPAA Parental Rating	1=G 2=PG 3=PG-13 4=R 5=NC-17 6=NR
4	ICRAParentalRatingNudityCS	Internet Content Rating Association Parental Rating for Nudity	1=Level 4 2=Level 3 3=Level 2 4=Level 1 5=Level 0 6=None
5	RIAAParentalRatingCS	RIAA Parental Rating	1=Parental advisory 2=None
6	ICRAParentalRatingSexCS	Internet Content Rating Association Parental Rating for Sex	1=Level 4 2=Level 3 3=Level 2 4=Level 1 5=Level 0 6=None

Rating _type	Name	Description	Rating_value
7	MPAAParentalRatingTVCS	MPAA Parental Rating for TV	1=TVY 2=TVY7 3=TVG 4=TVPG 5=TV14 6=TVMA 7=None
8	ICRAParentalRatingViolence		1=Level 4 2=Level 3 3=Level 2 4=Level 1 5=Level 0 6=None
9	GermanyFSK	German Freiwillige Selbstkontrolle der Filmwirtschaft Rating System	1=0 (Freigegeben ohne Altersbeschränkung) 2=6 (Freigegeben ab 6 Jahren) 3=12 (Freigegeben ab 12 Jahren) 4=16 (Freigegeben ab 16 Jahren) 5=18 (Keine Jugendfreigabe)

7.2.2.3 Copy_control_information access criteria descriptor

7.2.2.3.1 General

This descriptor represents the CCI information byte to the programme or service. It may optionally be used in networks that inject CCI, like the US cable. The descriptor tag for this descriptor is defined in Table 9. This descriptor is encoded as specified in Table 12.

Table 12 – Copy_control_information access criteria descriptor

Copy_control_information access criteria descriptor	Length	Type
copy_control_information_byte	8	bslbf

The copy_control_information_byte (a.k.a. CCI-byte) is a single byte (8-bit) field; see Table 13 and Table 14. Six of the eight bits are defined. The remaining two are reserved. The reserved bits shall be set to zero by the network.

NOTE It is out of the scope of this standard how to translate the received CCI value to a particular output protection technology (for example, HDCP). The copy_control_information access criteria descriptor will apply equally to all output protection technologies that are designed to respond to CCI. Some other output protection technologies may implement different content protection measures that do not easily map to CCI. If it is necessary to distinguish between different systems, the export permission in rights objects (see Clause A.21) can be used to signal the applicable rights individually for each system.

Table 13 – Bit assignments of copy_control_information_byte

Bit #	7	6	5	4	3	2	1	0
Network sets	0	0	RCT	CIT	APS1	APS0	EMI1	EMI0
DRM interprets as	rsvd	rsvd	RCT	CIT	APS1	APS0	EMI1	EMI0

Table 14 – CCI bit assignments

Short name	Description
EMI	Encryption mode indicator
APS	Analogue protection system
CIT	Constraint image trigger
RCT	Redistribution control trigger

7.2.2.3.2 EMI – DIGITAL COPY CONTROL BITS

The two least significant bits of the CCI byte are the EMI bits. They shall control copy permissions for digital copies. The EMI bits shall be supplied to any digital output ports for control of copies made from those outputs. The EMI bits are defined in Table 15.

Table 15 – EMI values and content

EMI value	Digital copy permission	Content type
00	Copying not restricted	Not "high value"
01	No further copying is permitted	High value
10	One-generation copy is permitted	High value
11	Copying is prohibited	High value

7.2.2.3.3 APS – ANALOG PROTECTION SYSTEM

Bits 3 and 2 of CCI as shown in Table 13 are the APS bits 1 and 0 respectively. The host shall use the APS bits to control copy protection encoding of analogue composite outputs as defined in Table 16.

Table 16 – APS value definitions

APS	Description
00	Copy protection encoding off
01	AGC process on, split burst off
10	AGC process on, 2-line split burst on
11	AGC process on, 4-line split burst on

7.2.2.3.4 CIT – CONSTRAINED IMAGE TRIGGER

A constrained image means the visual equivalent of not more than 520 000 pixels per frame (for example, an image with resolution of 540 vertical lines by 960 horizontal lines for a 16:9 aspect ratio). A constrained image can be output or displayed using video processing techniques such as line doubling or sharpening to improve the perceived quality of the image.

Bit 4 of CCI as shown in Table 13 is the CIT bit. The device shall use the CIT bit to control the image constraint of high-definition analogue component outputs as specified in Table 17.

Table 17 – CIT values and application

CIT value	Image constraint application
0	No image constraint asserted
1	Image constraint required

7.2.2.3.5 RCT – REDISTRIBUTION CONTROL TRIGGER

RCT is used to trigger the encryption plus non-assertion ("EPN") state in DTCP protected digital outputs in the DTCP licensed products.

Bit 5 of CCI as shown in Table 13 is the RCT bit. The device shall use the RCT bit as specified in Table 18 to trigger redistribution control on controlled content when the RCT value is set to a value of one (1) in combination with the EMI bits set to a value of zero, zero (0,0), which signals the need for redistribution control to be asserted on controlled content without the need to assert numeric copy control.

Table 18 – RCT values and application

RCT value	Redistribution control application
0	No redistribution control asserted
1	Redistribution control required

7.2.2.4 Blackout_spotbeam access_criteria_descriptor

7.2.2.4.1 General

This descriptor represents the black-out and/or spot beam to the programme or service. This descriptor may be used in networks that use black-out-spotbeam to limit access for a variety of reasons, for example, on geographical grounds.

The selective distribution of rights objects can be used to have fine-grained control over access. However, the blackout_spotbeam access_criteria_descriptor offers a mechanism that, depending on the situation, may achieve the same while using less bandwidth.

When using the black-out-spot-beam mechanism, devices are equipped during registration with a CGF (customer_group_filter, see 9.3.2.7.2). The bits in the CGF represent whether a device belongs to certain categories or not. The definition of these categories is outside the scope of this standard. Some examples are given in Clause B.9.

Furthermore, when using the black-out-spot-beam mechanism, the KSM may contain blackout_spotbeam access_criteria_descriptors, which in turn contain a black-out-spot-beam mask (bosb_mask). Subclause 7.2.2.4.3 defines the rules for denying access using the black-out-spot-beam mechanism.

When using the blackout_spotbeam descriptor, the COC (in call centre and/or in information in the purchase data) shall take care to avoid that customers purchase and pay for events where the end effect will be "no access".

7.2.2.4.2 Syntax and field definition of the blackout_spotbeam access_criteria_descriptor

The descriptor tag for this descriptor is defined in Table 9. This descriptor is encoded as specified in Table 19.

Table 19 – Blackout_spotbeam access criteria descriptor

Field	Length	Type
bosb_masklength	6	uimsbf
offset	6	uimsbf
operator	3	bslbf
invert_flag	1	bslbf
bosb_mask	bosb_masklength * 8	bslbf

bosb_masklength – This field represents the length in bytes of the bosb_mask carried in this access criteria descriptor. The value of this field times 8 defines how many black-out- spot-beam items are addressed (see 7.2.2.4.3 for the rules). The value of this field shall be equal to the length of the CGF from the keyset (see 9.3.2.7.2). The value of this field shall be greater than zero.

NOTE Choosing the length of the masks is the responsibility of the network operator and is out of scope for this standard. The granularity of a black-out/spot beam may differ in terrestrial, cable or satellite network conditions and is, therefore, a trade-off between payload overhead and limitations in expression. For example, a satellite network with a black-out-spot-beam model based on regions and cities might require a larger mask than the one for a terrestrial network. The length of a complete KSM should not exceed 255 bytes in a SimulCrypt environment to avoid repackaging before the SCS component.

offset – This field indicates the border in between the 'left' and the 'right' part of the bosb_mask. The offset field represents the offset in bytes from the least significant bit side (right side) of the bosb_mask field that indicates the end of the 'right' part and the start of the 'left' part of the bit mask. For example, offset 0x01 will indicate that the offset between the 'left' and the 'right' part is located between the right-most and one but right-most byte. See 7.2.2.4.3 for the rules of using this field and to Clause B.9 for examples. Table 20 defines the situations in which this field has an effect.

operator – This field indicates the operator that has to be applied; see Table 20. See 7.2.2.4.3 for the rules of using this field and Clause B.9 for examples.

Table 20 – Operator field values and their meaning

Operator field value	Mnemonic	Meaning	Offset field
0	BOSB_OR	OR of all bit-wise matches, whose result is equal to OR of left and right part match	Value is don't care
1	BOSB_AND	AND left and right part match	Value is used
2	BOSB_LEFT_OVER_RIGHT	Precedence of left part match over right part match	Value is used
3	BOSB_RIGHT_OVER_LEFT	Precedence of right part match over left part match	Value is used
4-7		Reserved for future use	

invert_flag – This 1-bit flag indicates whether the black-out-spot-beam output shall be inverted or not. See 7.2.2.4.3 for the rules of using this flag and Clause B.9 for examples.

bosb_mask – This field represents the mask to signal that devices are touched by a black-out or a spot beam. The bosb_mask is encoded as bit mask where any 0 means 'black-out' and any 1 mean 'spotbeam'. The bosb_mask can be divided into a 'left' part and a 'right' part, see the definition of the offset field above. See 7.2.2.4.3 for the rules of using the bosb_mask and Clause B.9 for examples.

7.2.2.4.3 Black-out-spot-beam access rules

The black-out-spot-beam mechanism by itself cannot give access to a service or programme. It can yield "bosb_access_denied" and "bosb_access allowed". The outcome of the black-out-spot-beam mechanism is then used according to the precedence rules; see Clause A.20.

There shall not be more than one blackout_spotbeam access criteria descriptor present in each KSM. The effect of the blackout_spotbeam access criteria descriptor is valid until the next KSM. If there is no blackout_spotbeam access criteria descriptor in a KSM, the consumption of the service or programme is not limited by the blackout_spotbeam mechanism until the next KSM.

The following rules define the outcome of the black-out-spot-beam mechanism.

```
// The output of the blackout spotbeam mecahnism is stored in the variable bosb_out.
// bosb_out can have the value 'bosb_access_denied' or 'bosb_access_allowed'

// When no CFG, so not registered, but KSM contains bosb_mask ➔ deny access.
if (bosb_mask in KSM && no CGF in registration data)  { bosb_out = bosb_access_denied; }

else { // Both bosb_mask and CGF are present ➔ match bosb_mask with CGF
  // Construct right and left parts
  bosb_mask_r = bosb_mask << (bosb_masklength - offset)*8;
  CGF_r       = CGF       << (bosb_masklength - offset)*8;
  // bosb_mask_r and CGF_r will become 0 when offset is equal to 0

  bosb_mask_l  = bosb_mask >> offset*8;
  CGF_l        = CGF       >> offset*8;

  if (operator==BOSB_LEFT_OVER_RIGHT) { // Left part has precedence
    if (CGF_l==0) { // Device is not part of any category in CGF_l ➔ use right parts
      if ( (bosb_mask_r & CGF_r) != 0) { output = match;   }
      else                             { output = no_match; }
    }
    else { // Use left parts for matching
      if ( (bosb_mask_l & CGF_l) != 0) { output = match;   }
      else                             { output = no_match; }
    }
  }

  else if (operator==BOSB_RIGHT_OVER_LEFT) { // Right part has precedence
    if (CGF_r==0) { // Device is not part of any category in CGF_r ➔ use left parts
      if ( (bosb_mask_l & CGF_l) != 0) { output = match;   }
      else                             { output = no_match; }
    }
    else { // Use right parts for matching
      if ( (bosb_mask_r & CGF_r) != 0) { output = match;   }
      else                             { output = no_match; }
    }
  }

  else if (operator==BOSB_OR) {  // OR operation
    if ( (bosb_mask & CGF) != 0)       { output = match;   }
    else                               { output = no_match; }
  }

  else if (operator==BOSB_AND) { // AND left and right part match
    if (((bosb_mask_l & CGF_l) != 0) &&
        ((bosb_mask_r & CGF_r) != 0))  { output = match;   }
    else                               { output = no_match; }
  }
  else                                 { output = no_match; }

  // Apply invert_flag
  if      ( (invert_flag==0) && (output==   match) ) { bosb_out = bosb_access_allowed;}
  else if ( (invert_flag==0) && (output==no_match) ) { bosb_out = bosb_access_denied; }
```

```
  else if ( (invert_flag==1) && (output==   match) ) { bosb_out = bosb_access_denied; }
  else                                               { bosb_out = bosb_access_allowed;}
}
```

7.2.3 Constants

Table 21 specifies the constants used in the key stream message.

Table 21 – Constants in key stream message

Name	Value
KSM_ALGO_IPSEC	0
KSM_ALGO_SRTP	1
KSM_ALGO_ISMACRYP	2
KSM_ALGO_MPEG2_TS_CRYPT	7
KSM_FLAG_FALSE	0
KSM_FLAG_TRUE	1
KSM_FLAG_EVEN	0
KSM_FLAG_ODD	1

7.2.4 Coding and semantics of attributes

protocol_version – This parameter indicates the protocol version of this key stream message. The device shall ignore messages that have a protocol_version number it does not support. The value of the protocol_version of this message shall be set to 0x1 (i.e. the original format).

If this parameter is set to 0x1, the format specified in this version of the standard is used. If this parameter is set to anything other than 0x1, the format is beyond the scope of this version of this standard.

reserved_for_future_use – These bits are reserved for future use and shall be set to 0 in systems according to this version of this standard.

traffic_protection_protocol – Defines the protocol used for the encryption and optional authentication of traffic:

KSM_ALGO_IPSEC = IPsec ESP (transport mode; encryption: AES-128-CBC [key length 128]; authentication: HMAC-SHA1-96 [key length 160] or NULL);

KSM_ALGO_SRTP = SRTP (encryption: AES-128-CTR [key length 128]; authentication: HMAC-SHA1-80 [key length 160] or NULL);

KSM_ALGO_ISMACRYP = ISMACryp (encryption: AES-128-CTR [key length 128]; authentication is not used);

KSM_ALGO_MPEG2_TS_CRYPT = DVB-CSA encryption or any other encryption algorithm listed in this subclause in the specification of the **content_key_index** field; authentication is not used;

other values = reserved for future use.

Whether or not authentication is used depends on <traffic_authentication_flag>.

traffic_authentication_flag – Defines whether or not the traffic is authenticated:

KSM_FLAG_FALSE = traffic authentication is not used;

KSM_FLAG_TRUE = traffic authentication is used, and the algorithm depends on <traffic_protection_protocol>.

next_traffic_key_flag – indicates whether or not the key stream message contains the next traffic key material:

KSM_FLAG_FALSE = the key stream message contains only the current traffic key material;

KSM_FLAG_TRUE = the key stream message contains both the current and the next traffic key material.

The next traffic key material shall be included at least 1 s (2 s if traffic_protection_protocol == KSM_ALGO_MPEG2_TS_CRYPT) before it becomes current. This is to enable the devices to process the traffic key material and put the necessary security associations in place before the media packets start arriving that are encrypted with the next traffic encryption key.

The next traffic key material shall not be included earlier than 1 min before it becomes current. This is to limit the effect on pay-per-view enforcement that is caused by sending the next traffic key material, encrypted with the encryption key of a programme that may end before the next traffic key becomes current, to maximally 1 min.

The above times shall be relative to the moment of transmission of the key stream messages.

timestamp_flag – Indicates whether or not the key stream message contains a timestamp:

KSM_FLAG_FALSE = the key stream message does not contain a timestamp;

KSM_FLAG_TRUE = the key stream message contains a timestamp.

programme_flag – Indicates whether or not the programme key layer is present in the key stream message:

KSM_FLAG_FALSE = the programme key layer is not present, i.e. the optional programme key layer is not used for the service;

KSM_FLAG_TRUE = the programme key layer is present, i.e. the optional programme key layer is used for the service.

<programme_flag> and <service_flag> shall not both be 0. All other combinations are allowed, indicating that either or both of the key layers are present.

service_flag – Indicates whether or not the service block is present in the key stream message:

KSM_FLAG_FALSE = the service key layer is not present, i.e. the optional service key layer is not used for the service;

KSM_FLAG_TRUE = the service key layer is present, i.e. the optional service key layer is used for the service.

<programme_flag> and <service_flag> shall not both be 0. All other combinations are allowed, indicating that either or both of the key layers are present.

security_parameter_index – Provides the link to the IPsec ESP header:

Upon reception of a protected IP packet, the device shall use the security parameter index (SPI) to identify (look up) the correct security association and thereby find the decryption and authentication keys to be used for the received IPsec ESP packet. The SPI value shall be in the range 0x00000100 – 0xFFFFFFFF. An incoming ESP packet containing the SPI value specified in this field shall use the key material provided in the encrypted traffic key material field as key material for the decryption operation.

next_security_parameter_index – Provides the link to the IPsec ESP header:

This field is present in the packet only if next traffic key flag is set to true. This field then contains the IPsec SPI value corresponding to the next_encrypted traffic key material field. The value of the SPI shall be in the range 0x00000100 – 0xFFFFFFFF. An incoming ESP packet containing the SPI value specified in this field shall use the key material provided in the next encrypted traffic key material field as key material for the decryption operation.

master_key_index_length – Provides the length of the master_key_index field.

This field gives the length of the master_key_index field in bytes.

master_key_index – Provides the link to the SRTP header:

Upon reception of a protected RTP packet, the device shall use the master key index (MKI) to identify (look up) the correct security association and thereby find the decryption and authentication keys to be used for a received SRTP packet.

This field is a sequence of 8-bit values. The sequence consists of master_key_index_length bytes. The bytes are in the same order that they will be in an SRTP packet and shall be in SRTP (see IETF RFC 3711) network byte-order when extracting the MKI value.

The MKI is associated with the current TEK. If the next traffic key flag is set to 1, the MKI associated with the "next TEK" is implicitly defined as MKI+1.

master_salt_flag – Specifies if the master salt is included in the SRTP parameters. For this version of this standard, this flag is always false, thus indicating that the master salt is set to a NULL value consisting of 112 0-bits.

key_indicator_length – The length of key_indicator for ISMACryp in bytes (1…255).

key_indicator – Is used by ISMACryp to associate ISMACrypKey with the access unit(s) of content encrypted with it.

content_key_index – Is used by MPEG2 TS CRYPT scrambling to identify the type of the MPEG2 TS CRYPT content key, so the device can take according measures. Table 22 defines the options.

Table 22 – Content_key_index options

Content_key_index value	Description	Comment
0x0	DVB-CSA key of 64-bit length.	
0x1	DES key of 56-bit length.	
0x2	3DES key of 168-bit length.	
0x3	3DES key of 112-bit length.	
0x4	3DES key of 56-bit length.	
0x5	AES key of 128-bit length.	
0x6	M2 key of 64-bit length.	Multi 2 for Japan
0x7 – 0xF	Reserved for future use.	

odd_even_flag – Indicates if the odd or even register is used for the content key, assuming the traffic_protection_protocol == KSM_ALGO_MPEG2_TS_CRYPT.

KSM_FLAG_ODD = the receiving device should insert the IEK (i.e. the content key) into the odd register of the DVB descrambler.

KSM_FLAG_EVEN = the receiving device should insert the IEK (i.e. the content key) into the even register of the DVB descrambler.

cipher_mode – Indicates the mode in which the cipher indicated by the field content_key_index is used, assuming the traffic_protection_protocol field has the value KSM_ALGO_MPEG2_TS_CRYPT. Table 23 defines the options.

Table 23 – cipher_mode options

cipher_mode value	Description	Comment
0x0	ECB	For DES, 3DES or AES
0x1	CBC	For DES, 3DES or AES
0x2	CSA	For DVB CSA
0x3 – 0x7	Reserved for future use	–

initial_vector_length – is the length in bytes of the initial vector.

initial_vector – is the initial vector for the current traffic_key_lifetime period that is used when, under KSM_ALGO_MPEG2_TS_CRYPT operation, the DES, 3DES or AES ciphers are used in CBC mode, as is indicated by the cipher_mode field. The value shall be a random number matching the size indicated by the field content_key_index (see Table 22).

next_initial_vector – is the initial vector for the next traffic_key_lifetime period that is used when, under KSM_ALGO_MPEG2_TS_CRYPT operation, the DES, 3DES or AES ciphers are used in CBC mode, as is indicated by the cipher_mode field. The value shall be a random number matching the size indicated by the field content_key_index (see Table 22).

NOTE 1 The initial vector is not used in ECB mode.

encrypted_traffic_key_material_length – Is the length in bytes of the encrypted traffic key material.

The length of the traffic key material depends on the encryption and authentication algorithm, and is obtained by adding the respective key sizes. Encryption may require the clear-text key material to be padded.

encrypted_traffic_key_material – Is the key material currently used for encryption and optional authentication of the traffic, encrypted using AES-128-CBC, with fixed IV 0, and with 0 padding in the last block, if needed.

If <programme_flag> == KSM_FLAG_TRUE, the traffic key material is encrypted with the programme encryption key (PEK).

If <programme_flag> == KSM_FLAG_FALSE and <service_flag> == KSM_FLAG_TRUE, the traffic key material is encrypted with the service encryption key (SEK).

After decryption (and discarding any padding), the traffic encryption key (TEK) and the traffic authentication key (TAK) are obtained in a way that depends on the protocol used for traffic protection.

a) IPsec:
If no traffic authentication is used, the TEK is identical to the decrypted traffic key material (16 bytes). If traffic authentication is used, TEK and traffic authentication seed (TAS) are

obtained by splitting the decrypted traffic key material into two parts, where the TEK is identical to the first 16 bytes, and the TAS is identical to the second 16 bytes. The TAK (20 bytes) is derived from the TAS, as specified in Clause A.8.

b) SRTP:
The master key is identical to the decrypted traffic key material and shall always be a 16-byte AES key as required by SRTP. SRTP specifies how to derive session encryption and authentication keys from the master key using a derivation function based on AES in counter mode.

c) ISMACryp:
No traffic authentication is used. TEK is identical to the decrypted traffic key material (16 bytes).
TEK is the binary representation of the "aes-key" part of the ISMACrypKey. The complete ISMACrypKey (as understood by an ISMACrypKey parser) can be constructed by the device using the ISMACrypSalt in the SDP file (see IETF RFC 2327) and the KSM parameter key_indicator as follows:

ISMACrypKey = (key)BASE64(TEK||ISMACrypSalt)|2^64|key_indicator

NOTE 2 In the above definition of ISMACrypKey, the character '|' is not the OR operation, as specified in 3.2 but simply means the insertion of the '|' character in the character string making up the ISMACrypKey.

The optional ISMA lifetime parameter is not signalled, assuming 2^64 (the default).

d) MPEG2 TS CRYPT:

If traffic_protection_protocol == KSM_ALGO_MPEG2_TS_CRYPT, the KSM can deliver a content key as specified in Table 24.

Table 24 – Obtaining the content key

Content_key_ index value	Description	Traffic key material	Obtain decrypted traffic key material
0x0	DVB-CSA key of 64-bit length	64 bits input are padded to 128 bits according to FIPS PUB 197:2001.	First 8 bytes
0x1	DES key of 56-bit length	56 bits input are padded to 128 bits according to FIPS PUB 197:2001.	First 7 bytes
0x2	3DES key of 168-bit length	168 bits input are padded to 256 bits according to FIPS PUB 197:2001.	First 21 bytes
0x3	3DES key of 112-bit length	112 bits input are padded to 128 bits according to FIPS PUB 197:2001.	First 14 bytes
0x4	3DES key of 56-bit length	56 bits input are padded to 128 bits according to FIPS PUB 197:2001.	First 7 bytes
0x5	AES key of 128-bit length	128 bits input are according to FIPS PUB 197:2001.	First 16 bytes
0x6	M2 key of 64-bit length	64 bits input are padded to 128 bits according to FIPS PUB 197:2001.	First 8 bytes
0x7 – 0xF	Reserved for future use	–	–

next_encrypted_traffic_key_material – Is the encrypted key material used for encryption and optional authentication of the traffic after the current crypto period is over and the next crypto period starts. The structure of this attribute is identical to encrypted_traffic_key_material attribute.

traffic_key_lifetime – denotes the lifetime of the traffic key material, relative to the first occurrence of an SPI, MKI or key_indicator.

If <traffic_key_lifetime> is n, then the actual lifetime is 2^n s, as presented in Table 25.

Table 25 – Traffic key lifetime

Value of traffic_key_lifetime attribute	0	1	2	3	4	5	6	7
Actual lifetime of traffic key material (seconds)	1	2	4	8	16	32	64	128

The actual duration of the crypto period shall be strictly shorter than the defined lifetime of the traffic key material. Typically, an SPI or MKI appears for the first time implicitly, when the "next" traffic key material is included in a KSM. Any safety margins to cope with network and transmission delays shall be added by the network. A typical value for the lifetime could be three times the crypto period.

The maximal value for the crypto period duration is in practice slightly shorter than the traffic key lifetime, because the KSM will include the "current" and "next" traffic key material before a change of crypto period, to allow the devices to set up the security associations.

After the lifetime has expired, the security association containing the traffic key can be safely deleted by the device. This may help managing the security association database in the device or enable other optimizations.

The maximum value for the traffic key lifetime is defined mainly in order to have a strict upper bound for the effect of the "sneak post view" problem: the "next traffic key" material is distributed under the current PEK, and allows viewers to view a programme during the next crypto period. Should this possibility still be of concern, the network may choose a shorter crypto period than the maximum value, or, during the crypto period where the current programme ends and a new programme starts, choose to distribute the "current" and the "next" traffic key material in separate KSMs, encrypted with their respective PEKs.

timestamp – Field containing a timestamp at the point of sending the key stream message. The timestamp shall be used as a reliable time of reception of the associated media stream for post-acquisition permissions. The device shall not use the timestamp as a reliable source for DRM time. The format of the 40-bit mjdutc field is specified in A.3.1.

access_criteria_flag – Indicates whether or not access criteria are defined for the programme:

KSM_FLAG_FALSE = no access criteria are defined. Access to the programme is governed by RO(s) associated with this programme or with the service this programme is a part of.

KSM_FLAG_TRUE = access criteria are defined, implying that the device is allowed to access the programme only if the specified access criteria are met and if the device has an RO granting access to the programme.

permissions_flag – Indicates whether or not permissions category is defined for the programme:

KSM_FLAG_FALSE = no permissions category is defined;

KSM_FLAG_TRUE = permissions category is defined.

number_of_access_criteria_descriptors – Indicates the number of access criteria descriptors.

permissions_category – Indicates the permissions category for the programme; see Table 26.

Table 26 – Values of permissions_category and their meaning

Value of permissions_ category	Meaning
0x00	No permissions category. In case of ICROs, ICROs with the following service_CID apply: service_CID as specified at the specification of the service_CID_extension field (see service_CID_extension), or service_CID = ”cid:” || socID || "#S" || serviceBaseCID || "@" || hex(service_CID_extension) || "_" || hex(0). In the case of BCROs, service RO assets with no permissions_category field or with permissions_category field 0 apply.
0x01...0x3F	**Permissions_category is included in the permissions lookup** In the case of ICROs, the device shall use as service_CID for permissions lookup the text string service_CID = ”cid:” || socID || "#S" || serviceBaseCID || "@" || hex(service_CID_extension) || "_" || hex(permissions_category) and then apply the permissions specified in the service ICRO for this asset. In the case of BCROs, the device shall look up the permissions specified in the service BCRO for the asset that has a matching permissions_category field. For both ICROs and BCROs, the permissions so found replace the service RO as specified in Clause A.20. In particular, if, for example, the so-found permissions do not include an ACCESS permission, the device does not have the right to render the broadcast stream immediately, as specified in the definition of the ACCESS permission.
0x40...0xFE	**Reserved for future standardization** If permissions_category is in the (reserved for future standardization) range 0x40...0xFF, and the device does not support it, the device shall drop (i.e., ignore) all permissions (like play, redistribute, etc.) indicated in the service RO, or if the device cannot do such permission dropping, allow real-time rendering of the streaming content only (i.e. refuse to record the content, or to redistribute it in real time). Permissions_category has no impact on a programme RO. The permissions delivered in a programme RO apply as such; see Clause A.20.
0xFF	**No post-acquisition content protection** Export in plaintext is allowed.

encrypted_PEK – Is the programme encryption key (PEK) used within the current key stream message to decrypt the traffic key material, encrypted using AES-128-CBC with fixed IV 0).

The programme encryption key is encrypted with the service encryption key (SEK).

programme_CID_extension – Is the extension of the programme_CID that allows to identify the PEAK that has been delivered to the device within a programme RO.

The CID/BCI of the programme key is constructed as

```
programme_CID = "cid:" || socID || "#P" || serviceBaseCID || "@" ||
hex(programme_CID_extension)
programme_BCI = hash("cid:" || socID || "#P" || serviceBaseCID || "@")
|| programme_CID_extension
```

The socID and serviceBaseCID are string values and are expected to be part of the service guide, see e.g. A.17, A.18 and A.19. Upon reception of a KSM, the device can assemble the programme_CID/BCI and look up the programme key (wrapped inside an RO). The device should check whether it has a programme RO for the programme_CID/BCI.

The hex() function is a hexadecimal presentation of the parameter containing hexadecimal characters 0-9 and a-f (in lower case) with possible preceding zeros.

EXAMPLE For a 16-bit value 2748, hex() returns "0abc". There are always two characters generated for each byte.

The hash function for the construction of programme_BCI is defined in Clause A.4, where BCI is defined. It does not depend on the contents of the KSM, and can thus be pre-computed.

programme_MAC – Is the HMAC-SHA-1-96 according to IETF RFC 2104 and IETF RFC 2404 calculated over all preceding fields of the key stream message. It is used to authenticate the relevant part of the key stream message with PAK in case of pay-per-view, where a PEK from a programme RO is used to decrypt the traffic key material directly.

In case the device is accessing the key stream message with a programme RO, the device shall compute the programme MAC, and drop the message if authentication fails. In this case, <programme_MAC> may also be used to detect and drop duplicates (it can be expected that a particular key stream message is repeated multiple times, in order to keep access times short for devices that newly start receiving a broadcast transmission).

In case the device is accessing the key stream message with a service RO, it will not be able to compute the programme MAC, and there is no need for it to do so.

service_CID_extension – Is the extension of the service_CID that allows the identification of the SEAK that has been delivered to the device within a Service RO.

The CID/BCI of the service key is constructed as:

```
service_CID = "cid:" || socID || "#S" || serviceBaseCID || "@" ||
hex(service_CID_extension)
service_BCI = hash("cid:" || socID || "#S" || serviceBaseCID || "@") ||
service_CID_extension
```

The socID and serviceBaseCID are string values and are expected to be part of the service guide, see e.g. A.17, A.18 and A.19. Upon reception of a KSM, the device can assemble the service_CID/BCI and look up the service key (wrapped inside an RO). The device should check whether it has a service RO for the service_CID/BCI.

The hex() function is a hexadecimal presentation of the parameter containing hexadecimal characters 0-9 and a-f (in lower case) with possible preceding zeros.

EXAMPLE For a 16-bit value 2748, hex() returns "0abc". There are always two characters generated for each byte.

The hash function for the construction of service_BCI is defined in Clause A.4, where BCI is defined. It does not depend on the contents of the KSM and can thus be pre-computed.

If the permissions_category field is present and has a non-zero value, the service_CID of the service is constructed as specified above (at the specification of the permissions_category field).

service_MAC –Is the HMAC-SHA-1-96 according to IETF RFC 2104 and IRTF RFC 2404 calculated over all preceding fields of the key stream message. It is used to authenticate the key stream message with SAK in case of subscription, where a SEK from a service RO is used to decrypt the PEK and further decrypt the traffic key material.

In case the device is accessing the key stream message with a service RO, the device shall compute the service MAC and drop the message if authentication fails, i.e. if the computed MAC does not correspond to <service_MAC>. In this case, <service_MAC> may also be used to detect and drop duplicates (it can be expected that a particular key stream message is repeated multiple times, in order to keep access times short for devices that newly start receiving a broadcast transmission).

In the case that the device is accessing the key stream message with a programme RO, it need not compute the service MAC.

8 Rights management layer

8.1 General

In this clause, the rights management layer is specified. The rights management layer is layer 3 of the four-layer model (see 5.4) used in the protection functionality specified by this standard. The specific use of the rights management layer for the individual supported systems is further specified in 13.4, 14.4, 15.5 and 16.4.

8.2 Identification of rights objects

In the case of a subscription, the SEAK associated with the service is delivered to the authorized device in an RO that is bound to a device, a unique group, a subscriber group or to a domain. Such an RO is called a service RO. In general, a service RO will contain key material associated with more than one service (with a service bundle).

In the case of pay-per-view, the PEAK associated with a pay-per-view event is delivered to the authorized device direct within an RO that is bound to a device, a unique group, a subscriber group or to a domain. Such an RO is called a "programme RO".

The ID of ROs that contain SEAKs or a PEAK needs to be structured, to allow for the management of purchase transactions in the device, or more specifically, to create an association between the service guide (where the purchase item is expected to be announced) and the successful completion of the purchase transaction (when the RO related to the purchase has finally been received in the device). This is valid for both interactive and especially for broadcast mode of operation, where the RO may be received by the device much later than the purchase transaction is initiated.

Defining a structured ID will allow the device also to check later on whether ROs for all subscribed services are available (and have been renewed). The rekeying_period_number is an increasing number by which the roID related to the same purchase item can be made unique.

The ID of an OMA DRM 2.0 RO delivered over an interactivity channel (i.e. the ID of an ICRO), linked with a subscription (service RO) or pay-per-view (programme RO) and bound to a device or to a domain, shall be constructed respectively as follows:

```
deviceRoID = "E" || deviceID || "_S" || socID || "_I" ||
purchase_item_id || "_" || hex(rekeying_period_number)
```

```
domainRoID = "O" || domainID || "_S" || socID || "_I" ||
purchase_item_id || "_" || hex(rekeying_period_number)
```

The hex() function is a hexadecimal presentation of the parameter containing hexadecimal characters 0-9 and a-f (in lower case) with possible preceding zeros.

EXAMPLE For a 16-bit value 2748, hex() returns "0abc". There are always two characters generated for each byte.

In the case of BCROs, the link with the corresponding subscription (service RO) or pay-per-view (programme RO) is obtained by using the field purchase_item_id and rekeying_period_number (see 8.4.3).

8.3 Requirements for rights objects

8.3.1 Requirements for service ROs

A service RO shall contain at least one asset, that is one (service_period_CID/BCI, wrapped_SEAK) pair. The CID/BCI pertaining to a particular re-keying period of the service shall be constructed as specified in Clause A.4.

For BCROs, after unwrapping the SEAK contained in the RO the service encryption key (SEK) and the service authentication seed (SAS) are obtained by splitting the unwrapped key material into two parts as follows:

SEK = first part (128 bits, since AES-128 is used to encrypt the traffic key material or the PEK);

SAS = second part (128 bits).

For ICROs, SEK and SAS are carried as separate elements in the asset, as specified in A.8.3.2.

The SAK (160 bits, since HMAC-SHA-1-96 is used to calculate the service_MAC) is derived from the SAS, as specified in Clause A.8.

The service RO shall contain a rights expression defining a "date-time" constraint for the "access" permission. The time interval defined by this constraint shall correspond to the time interval during which the SEAK can be used to get access to the key stream message. The device should use this information in order to determine the appropriate time for re-keying the service RO.

All SEAKs that are bundled in a single service RO shall have the exact same lifetime, and the "access" permission shall be applicable to all SEAKs. This allows the device to determine the time for RO renewal in an unambiguous way.

If and only if post-delivery content protection is used in a system, the service RO may contain further rights expressions restricting post-acquisition usage, and such restrictions shall be enforced.

The permissions may be different for different programmes, depending on the permissions_category value signalled in KSM (see Clause 7). The service RO shall specify (as separate assets) the permissions for each of the categories in the range 0x01...0x3F that are used in the service.

The "access" permission is a new extension to OMA DRM for the purpose of defining rights to service protection in a clean manner that is distinguishable from usage rules defined for content protection. For maximum compatibility with older OMA DRM implementations, it is recommended that the "execute" permission be granted as well.

8.3.2 Requirements for programme ROs

A programme RO shall contain exactly one asset, that is a (programme_CID/BCI, wrapped_PEAK) pair. The CID/BCI pertaining to the programme shall be constructed as specified in Clause A.4.

For BCROs, after unwrapping the PEAK contained in the RO, the programme encryption key (PEK) and the programme authentication seed (PAS) are obtained by splitting the unwrapped key material into two parts as follows:

PEK = first part (128 bits, since AES-128 is used to encrypt the traffic key material)

PAS = second part (128 bits)

For ICROs, PEK and PAS are carried as separate elements in the asset, as specified in A.8.3.2.

The PAK (160 bits, since HMAC-SHA-1-96 is used to calculate the programme_MAC) is derived from the PAS, as specified in Clause A.8.

The programme RO shall contain a rights expression defining a "date-time" constraint for the "access" permission. The time interval defined by this constraint shall correspond to the time interval during which the PEAK can be used to get access to the key stream message.

If and only if post-delivery content protection is used in a system, the programme RO may contain further rights expression restricting post-acquisition usage, and such restrictions shall be enforced.

The "access" permission is a new extension to OMA DRM for the purpose of defining rights to service protection in a clean manner that is distinguishable from usage rules defined for content protection. For maximum compatibility with older OMA DRM implementations, it is recommended that the "execute" permission be granted as well.

8.4 Format of rights objects

8.4.1 Format of an interactivity channel rights object (ICRO)

This subclause specifies the format of a rights object that can be obtained over an interactivity channel. Rights objects delivered over an interactivity channel are referred to as interactivity channel rights objects (ICRO). An ICRO is the OMA DRM V2 RO, as specified in OMA-ERP-DRM-V2_0, with extensions specified in 8.2 and A.8.3.2.

8.4.2 Format of a broadcast rights object (BCRO)

This subclause specifies the format of a rights object that can be distributed over a broadcast network.

Table 27 defines the format of a BCRO. The *asset*, *permission* and *constraint* object correspond in their meaning to their counterparts in OMA-ERP-DRM-V2_0. The *action* object corresponds to the allowed elements in the OMA-TS-DRM-REL-V2_0 permissions element.

BCROs from one rights issuer are normally carried in one stream, and this stream does only contain BCROs from this one rights issuer. As the rights issuer id is thereby already given by the BCRO stream, the BCRO does not have to carry this information. If, however, BCROs from different rights issuers have to be carried in one stream then the rights issuer id shall be signalled in every BCRO. This is done by setting the rights_issuer_flag to 1.

Table 27 – Format of BCRO

Field	Length	Type
BCRO() {		
/* MAC protected part starts here */		
message_tag	8	uimsbf
version	4	uimsbf
bcro_length	12	uimsbf
group_size_flag	1	bslbf
timestamp_flag	1	bslbf
stateful_flag	1	bslbf

Field	Length	Type
refresh_time_flag	1	bslbf
address_mode	3	uimsbf
rights_issuer_flag	1	bslbf
address	32	uimsbf
if(address_mode == 0x1 && group_size_flag == 0){		
bit_access_mask	256	bslbf
}else if(address_mode == 0x1 && group_size_flag == 1){		
bit_access_mask	512	bslbf
}else if (address_mode&0x6 == 0x2){		
position_in_group	8	uimsbf
}else if (address_mode == 0x4){		
domain_id_extension	6	bslbf
domain_generation	10	uimsbf
}		
if(rights_issuer_flag == 1){		
rights_issuer_id	160	bslbf
}		
if(timestamp_flag == 1){		
bcro_timestamp	40	mjdutc
}		
if(refresh_time_flag == 1){		
refresh_time	40	mjdutc
}		
permissions_flag	1	bslbf
rekeying_period_number	7	uimsbf
purchase_item_id	32	uimsbf
number_of_assets	8	uimsbf
for(i=0;i<number_of_assets; i++){		
asset()		
}		
if(permissions_flag == 1){		
number_of_permissions	8	uimsbf
for(i=0;i<number_of_permissions; i++){		
permission()		
}		
}		
/* MAC protected part ends here */		
MAC	96	bslbf
}		

message_tag – Tag identifying this message as a BCRO. The value for this field is defined in Clause A.12.

version – 4-bit flag that indicates the version of the BCRO message format. If this flag is set to 0, the original format is used. Devices shall ignore BCROs with versions it does not support.

bcro_length – 12-bit field indicating the length in bytes of the BCRO starting immediately after this field. The size of a BCRO including its header shall not exceed 4 096 bytes.

group_size_flag – 1-bit field indicating the group size used. 0 – a maximum group size 256 is used, 1 – a maximum group size of 512 is used.

timestamp_flag – 1-bit field indicating that the BCRO is time stamped.

stateful_flag – 1-bit flag indicating that when set to 1 the BCRO contains stateful information.

refresh_time_flag – 1-bit flag indicating that a refresh_time for the BCRO is contained in this BCRO.

address_mode – 3-bit field indicating the addressing mode used by this BCRO. Table 28 lists all four possible address modes.

Table 28 – Address_mode

Address_mode	Description
0x0	Addressing whole of unique group
0x1	Addressing of subscriber group using a bit_mask size of 256 or 512 bit depending on group_size_flag (subset of unique group)
0x2-0x3	Addressing of unique device
0x4	Addressing of OMA domain. Address field concatenated with the domain_id_extension will be the domain id in this case
0x5-0x7	Reserved

rights_issuer_flag – 1-bit flag indicating that the rights issuer id is listed in this BCRO. Normally, this information is given via a dedicated BCRO stream. This flag will only be set if BCROs from different rights issuers are carried in the same stream.

address – 4-byte group address. Each rights issuer has its own address space. If the group_size is 512 then the group address is made of the first 31 bits of the address field. If the BCRO is addressed to a unique device in a group, then the least significant bit of the address field is the most significant bit of the group position.

If the address_mode is set to 0x4, the address field contains the first 32 bits of the short form domain_id.

bit_access_mask – If the BCRO addresses a subset of a unique group (address_mode 0x1) than the bit_access_mask defines to which devices in the group this BCRO is addressed to. Devices not listed in the bit_access_mask cannot decrypt the key material in this BCRO as zero message broadcast encryption is used for the encryption of the key material. The size of the bit_access_mask is given by the group_size_flag.

position_in_group – If the BCRO addresses a unique device then this field specifies the position of the unique device in the given subscriber group. If group_size_flag is 0 than the position in the group is directly given by the position_in_group field. If group_size_flag is 1 then 9 bits are used to identify the position in the group. If group_size_flag is 1 then the lsb (bit 0) from the address field is used as the 9th bit, the most significant bit. The real position in the group is then given by:

```
int real_position_in_group;
if(address_mode&0x6==0x2){
```

```
  if(group_size_flag == 0){
    //maximum size of 256 devices in group.
    real_position_in_group = position_in_group;
  }else{
    //maximum size of 512 devices in group;
    real_position_in_group = ((address&0x1)<<8)|position_in_group;
  }
}
```

domain_id_extension – The domain_id is given by the address field concatenated with the domain_id_extension to form a 38-bit id:

```
domain_id = (address<<6)|domain_id_extension
```

domain_generation – This 10-bit field specifies the generation of the domain. The shortform_domain_id is constructed as follows: shortform_domain_id = (domain_id <<10) | domain_generation.

rights_issuer_id – The ID of the rights issuer. This is the 160-bit SHA-1 hash of the public key of the RI. See X509SPKIHash in OMA-TS-DRM-DRM-V2_0.

bcro_timestamp – Field containing a timestamp at the point of issuing of the BCRO. The format of the 40-bit mjdutc field is specified in A.3.1. This 40-bit field contains the timestamp of the BCRO in universal time, coordinated (UTC) and modified julian date (MJD).

refresh_time – The refresh_time specifies the time when the device should acquire a new BCRO. It does not specify when the keys in the BCRO expire. This field is a hint to a device to acquire a new BCRO for the content listed in the BCRO before the keys in the BCRO expires. The format of the 40-bit mjdutc field is specified in A.3.1. This 40-bit field contains the expiry time of the BCRO in universal time, coordinated (UTC) and modified julian date (MJD).

permissions_flag – 1-bit flag indicating that the BCRO contains at least 1 permission.

rekeying_period_number – 7-bit counter used to differentiate between different ROs with the same purchase_item_id.

purchase_item_id – 32-bit field specifying the purchase ID this RO is associated with.

number_of_assets – This field specifies the number of assets (see below) in this BCRO. Each asset listed in this BCRO has an internal id that is equal to the index of the asset in this BCRO. In other words, the first asset listed in this BCRO has the internal asset id (index) of 0, the second of 1 etc. This internal id or index is used by permissions objects (see below) to identify the assets it addresses.

asset() – This object is specified in 8.4.3.

number_of_permissions – This field specifies the number of permissions (see below) in this BCRO.

permission() – This object is specified in 8.4.4.

MAC – This is the authentication code calculated over all bytes before this field in the BCRO using HMAC-SHA-1-96, see IETF RFC 2104. The MAC is used for integrity check of the BCRO. The key used to create the MAC is the BCRO authentication key BAK as defined in A.8.4).

8.4.3 Format of the asset object

The format of the asset object is specified in Table 29.

Table 29 – Asset format

Field	Length	Type
asset() {		
BCI	96	bslbf
key_flag	1	
key_type	1	uimsbf
reserved_for_future_use	2	bslbf
inherit_flag	1	uimsbf
asset_type	2	uimsbf
permissions_category_flag	1	uimsbf
if(inherit_flag == 1){		
purchase_item_id	32	uimsbf
reserved_for_future_use	1	bslbf
rekeying_period_number	7	uimsbf
}		
if(permissions_category_flag == 1){		
permissions_category	8	uimsbf
}		
if(key_flag == 1){		
if(asset_type == 0){		
if(key_type == 0){		
encrypted_service_encryption_authentication_key	256	bslbf
}else if (key_type == 1){		
encrypted_program_encryption_authentication_key	256	bslbf
}		
}else if(asset_type == 0x1){		
encrypted_content_encryption_key	128	bslbf
}		
}		
}		

BCI – This 96-bit field is the binary content ID, see Clause A.4. A BCRO can contain multiple assets with the same BCI but with a different permissions_category. Only one asset has to carry key material.

key_flag – 1-bit flag indicating that the asset does contain key material.

key_type – 1-bit flag indicating the type of the key material. If set to 0 the key material contains a service encryption key (SEK), when set to 1 it contains a programme encryption key (PEK).

reserved_for_future_use – These bits are reserved for future use and shall be set to 0 in systems according to this version of this standard.

inherit_flag – 1-bit flag indicating whether inheritance is used. If this flag is set to 1, the asset inherits the rights setting from a parent rights object.

asset_type – 2-bit flag indicating the asset type as defined in Table 30. If the asset_type is set to 0, the asset may contain either a PEK or a SEK. If the asset_type is set to 0x1 then the asset may contain a CEK.

Table 30 – Asset_type

Asset_type	Description
0x0	Broadcast stream protected IPsec, SRTP, ISMACryp, or MPEG2 TS CRYPT as defined in this standard
0x1	Downloaded file content as defined by OMA
0x2-0x3	Reserved

permissions_category_flag – 1-bit flag indicating that a permissions_category field is present in this asset object.

purchase_item_id – 32-bit id specifying the purchase ID of the parent rights object.

rekeying_period_number – 7-bit field specifying the rekeying_period_number of the parent rights object. The purchase_item_id and rekeying_period_number are used together with the socID and deviceID or domainID to identify the parent rights object uniquely.

permissions_category – For programme assets, the value of this field (if present) is always zero. For service assets, the following rule applies. If the value of this field is non-zero, it indicates that the permissions (see below) linked to this asset are only to be applied for streaming content whose KSM contains the same value in its permissions_category field. If the value of this field is zero, it indicates that the permissions (see below) linked to this asset are only to be applied for streaming content whose KSM contains the value zero in its permissions_category field, or has value zero for its permissions_flag bit (indicating that there is no permissions_category field in the KSM). There may be multiple assets with the same Service_BCI, in which case typically only one of them contains authentication and/or encryption keys in it asset object(s). KSM permissions_category field value thus selects the one with the permissions to be applied among the service assets with the same Service_BCI. The one with the authentication and/or encryption keys is found among the BCROs via inheritance, or by lookup for a BCRO with key material in its assets.

encrypted_service_encryption_authentication_key – If key_type is set to 0, this field contains the encrypted SEAK, the service encryption key (SEK) concatenated with the service authentication seed (SAS); see also A.8.3. The field itself is protected using AES-128-CBC, with fixed IV 0 and with 0 padding in the last block if needed. The key used to decrypt this field depends on the addressing mode of the BCRO; see Table 31.

Table 31 – Mapping of address_mode to keys

Address_mode	Keys used
0x0 (unique group)	UGK (unique group key)
0x1 (subscriber group)	Deduced SEK decryption key (based on bit_access_mask and zero message subscriber group keys)
0x2 or 0x3 (unique Device)	UDK (unique device key)
0x4 (OMA domain)	BDK (broadcast domain key)

encrypted_program_encryption_authentication_key – If key_type is set to 1, this field contains the encrypted PEAK, the PEK concatenated with the PAS, see also A.8.3. The field itself is protected using AES-128-CBC, with fixed IV 0 and with 0 padding in the last block if needed. The key used to decrypt this field is depending on the addressing mode of the BCRO; see Table 32.

Table 32 – Mapping of address_mode to keys

Address_mode	Key(s) used to decrypt field
0x0 (unique group)	UGK (unique group key)
0x1 (subscriber group)	Deduced PEK decryption key (based on bit_access_mask and zero message subscriber group keys)
0x2 or 0x3 (unique Device)	UDK (unique device key)
0x4 (OMA domain)	BDK (broadcast domain key)

encrypted_content_encryption_key – This field contains the encrypted content encryption key (CEK). The field is protected using AES-128-CBC, with fixed IV 0 and with 0 padding in the last block if needed. The key used to decrypt this field is depending on the addressing mode of the BCRO; see Table 33.

Table 33 – Mapping of address_mode to keys

Address_mode	Key(s) used to decrypt field
0x0 (unique group)	UGK (unique group key)
0x1 (subscriber group)	Deduced CEK decryption key (based on bit_access_mask and zero message subscriber group keys)
0x2 or 0x3 (unique device)	UDK (unique device key)
0x4 (OMA domain)	BDK (broadcast domain key)

8.4.4 Format of the permission object

The format of the permission object is specified in Table 34.

Table 34 – Permission format

Field	Length	Type
permission() {		
constraint_flag	1	uimsbf
actions_flag	1	uimsbf
number_of_assets	6	uimsbf
for(i = 0;i<number_of_assets; i++){		
asset_index	8	uimsbf
}		
if(constraint_flag == 1){		
constraint()		
}		
if(actions_flag == 1){		
number_of_actions	8	uimsbf
for(i=0;I<number_of_actions; i++){		
action()		
}		
}		
}		

constraint_flag – 1-bit flag that indicates when set to 1 that a constraint object is present in this permissions object. The constraint object applies to all action listed in this permission object.

actions_flag – 1-bit flag. When set to 1, 1 or more actions are contained in this permission object.

number_of_assets – The number of assets this permission object links to. Assets linked to by this permission object are bound by this permission object.

asset_index – A list of number_of_assets links to assets in this BCRO. Assets are linked to by using the internal asset id (the index of the asset in this BCRO).

constraint () – This object is specified in 8.4.6.

number_of_actions – Field specifying the number of actions (see below) contained in this permission object.

action () – This object is specified in 8.4.5.

8.4.5 Format of the action object

The format of the action object is specified in Table 35.

Table 35 – Action format

Field	Length	Type
action() {		
action_type	7	uimsbf
constraint_flag	1	uimsbf
if(constraint_flag == 1){		
constraint()		
}		
}		

action_type – 7-bit field specifying the type of action as listed in Table 36.

Table 36 – Action_type

Action_type	Description
0x00	PLAY_ACTION
0x01	DISPLAY_ACTION
0x02	EXECUTE_ACTION
0x03	PRINT_ACTION
0x04	EXPORT_ACTION
0x05	ACCESS_ACTION
0x06	SAVE_ACTION
0x07-0x7F	Reserved for future use.

The action_types 0x00 (PLAY_ACTION), 0x01 (DISPLAY_ACTION), 0x02 (EXECUTE_ACTION), 0x03 (PRINT_ACTION) and 0x04 (EXPORT_ACTION) and their semantics are defined in OMA-TS-DRM-REL-V2_0.

The action_type 0x06 (SAVE_ACTION) and its semantics is defined in Clause A.24.

The action_type 0x05 (ACCESS_ACTION) and its semantics are defined below.

The ACCESS_ACTION grants the permission to create a transient representation of audio or video content direct from a broadcast stream during its reception. It contains an optional <constraint> object. If the constraint_descriptor is specified, the device shall grant access rights according to the constraint_descriptor child object. If no constraint_descriptor is specified, the device shall grant unlimited access rights.

A system_constraint object contained in an ACCESS_ACTION object is used to specify a target system that may be used for creating a transient rendering of the broadcast stream.

The ACCESS_ACTION has the semantics of rendering the broadcast stream into transient audio/video form, for example, audio/midi, video/quicktime. The device shall not grant access according to an ACCESS_ACTION to content that cannot be rendered in this way.

The device shall not grant access to stored content, not even stored broadcast streams, based on the ACCESS_ACTION. In order to specify rights for stored content, the PLAY_ACTION shall be utilized instead.

Clause A.23 specifies the use of the ACCESS_ACTION in ICROs.

constraint_flag – 1-bit flag that indicates when set to 1 that a constraint object is present in this action object. The constraint object only applies to the action it is in.

8.4.6 Format of the constraint object

8.4.6.1 Top-level format

The format of the constraint object is specified in Table 37.

Table 37 – Constraint format

Field	Length	Type
constraint() {		
number_of_constraints	4	uimsbf
constraint_descriptor_length	12	uimsbf
for(i=0;i;<number_of_constraints; i++){		
constraint_descriptor()		
}		
}		

number_of_constraints – 4-bit number specifying the number of constraint descriptors (see below).

constraint_descriptor_length – Length of all constraint descriptors in bytes that follow this field.

constraint_descriptor() – The format of this descriptor is specified in Table 38.

Table 38 – Format of constraint_descriptor

Field	Length	Type
constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
for(i=0;i;<length; i++){		
byte	8	uimsbf
}		
}		

constraint_tag – Tag identifying the specific constraint_descriptor as listed in Table 39.

Table 39 – Constraint_tag

Constraint_tag	Description
0x00	Count constraint
0x01	Timed-count constraint
0x02	Date time constraint
0x03	Interval constraint
0x04	Accumulated constraint
0x05	Individual constraint
0x06	System constraint
0x07	Token management constraint
0x08-0xFF	Reserved for future use

8.4.6.2 Count constraint descriptor

The format of the count_constraint_descriptor is specified in Table 40.

Table 40 – Format of count_constraint_descriptor

Field	Length	Type
count_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
count	8*length	uimsbf
}		

length – The number of bytes used for the count field. Length shall not exceed 4; hence the maximum size of the count field can be 32 bits.

count – The number of times the content can be played. The field can be of size 8, 16, 24 and 32 bits. See also OMA-TS-DRM-REL-V2_0.

8.4.6.3 Timed count constraint descriptor

The format of the timed_count_constraint_descriptor is specified in Table 41.

Table 41 – Format of timed_count_constraint_descriptor

Field	Length	Type
timed_count_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
timer	16	uimsbf
count	8*(length-2)	uimsbf
}		

length – The number of bytes following this field. The count field is length-2 bytes long and shall not exceed 32 bits.

timer – Specifies the number of seconds after which the count state is reduced starting from beginning to render the content.

count – The number of times the content can be played. The field can be of size 8, 16, 24 and 32 bits. See also OMA-TS-DRM-REL-V2_0.

8.4.6.4 Date-time constraint descriptor

The format of the date-time constraint descriptor is specified in Table 42.

Table 42 – Format of datetime_constraint_descriptor

Field	Length	Type
datetime_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
start_flag	1	bslbf
end_flag	1	bslbf
reserved_for_future_use	6	bslbf
if(start_flag == 1){		
start_time	40	mjdutc
}		
if(end_flag == 1){		
end_time	40	mjdutc
}		
}		

length – The number of bytes of the descriptor immediately following this field.

start_flag – 1-bit field. When set, the descriptor contains a start time.

end_flag – 1-bit field. When set, the descriptor contains an end time.

reserved_for_future_use – These bits are reserved for future use and shall be set to 0 in systems according to this version of this standard.

start_time – Time field with the semantics of ‘not before’ time for a permission. The start_time shall be before the end_time if present. See also OMA-TS-DRM-REL-V2_0. The format of the 40-bit mjdutc field is specified in A.3.1.

end_time – Time field with the semantics of ‘not after’ time for a permission. The end_time shall be after the start_time if present. See also OMA-TS-DRM-REL-V2_0.The format of the 40-bit mjdutc field is specified in A.3.1.

8.4.6.5 Interval constraint descriptor

The format of the interval_constraint_descriptor is specified in Table 43.

Table 43 – Format of interval_constraint_descriptor

Field	Length	Type
interval_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
time_interval	8*length	uimsbf
}		

length – The number of bytes following this field. Length specifies the size of the time_interval field.

time_interval – Specifies the number of seconds starting from first receiving this BCRO that the permission is valid. The length of the field is given by the length field and shall not exceed 32 bits. See also OMA-TS-DRM-REL-V2_0 for more details.

8.4.6.6 Accumulated constraint descriptor

The accumulated_constraint_descriptor specifies the maximum period of metered usage time during which the rights can be exercised over the DRM content. The format of the accumulated_constraint_descriptor is specified in Table 44.

Table 44 – Format of accumulated_constraint_descriptor

Field	Length	Type
accumulated_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
accumulated_time	8*length	uimsbf
}		

length – The number of bytes following this field. Length specifies the size of the accumulated_time field.

accumulated_time – Specifies the maximum period of metered usage time during which the rights can be exercised. The period is given in seconds. The length of the field is given by the length field and shall not exceed 32 bits. See also OMA-TS-DRM-REL-V2_0 for more details.

8.4.6.7 Individual constraint descriptor

The individual constraint is a constraint that is used to bind content to individuals. If the content should be bound to more than one individual, multiple individual_constraint_descriptor(s) can be carried in one constraint object. The format of the individual_constraint_descriptor is specified in Table 45.

Table 45 – Format of individual_constraint_descriptor

Field	Length	Type
individual_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
reserved_for_future_use	4	bslbf
id_type	4	uimsbf
individual_id	(length-1)*8	bslbf
}		

length – The number of bytes following this field. Length-1 specifies the size of the individual_id field.

reserved_for_future_use – These bits are reserved for future use and shall be set to 0 in systems according to this standard.

id_type – Tag identifying format of the individual_id as listed in Table 46.

Table 46 – Id_type

Id_type	Description
0x0	The individual_id field contains the IMSI number coded as 16-digit 4-bit BCD. The first digit shall be 0 and shall be ignored. The length of the individual_id field is 64 bit
0x1	The individual_id field contains the PKC-ID of the WIM to which the content is bound
0x2-0xF	Reserved for future use

individual_id – Individual ID. The format and length of this field is identified by the identifier_type and length field see the table above. See also OMA-TS-DRM-REL-V2_0 for more details.

8.4.6.8 System constraint descriptor

The system constraint is a constraint that is used to identify systems to which the content and rights objects are allowed to be exported to. The format of the system_constraint_descriptor is specified in Table 47.

Table 47 – Format of system_constraint_descriptor

Field	Length	Type
system_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
system_id	64	bslbf
parameterbytes	8*length - 64	bslbf
}		

length – The number of bytes following this field.

system_id – The system id of the system the content and RO can be exported to. This is the HMAC-SHA-1-64 encoded hash (using 0 as the key) of the system name as registered with OMNA. See also OMA-TS-DRM-REL-V2_0 for more details.

parameterbytes – This is a string of bytes, containing parameters for the system. This may be required when exporting to a (possibly non-DRM) system and requiring that no more copies are to be made. The format of parameters is system specific and out of scope of this standard.

8.4.6.9 Token management constraint descriptor

The token_management_constraint_descriptor specifies that the consumption of the DRM content involves the consumption of tokens. The device can receive tokens from each rights issuer and store them per rights issuer in a token store. The parameters in the token_management_constraint_descriptor indicate how "much" consumption of DRM content requires how many tokens to be consumed from the token store.

The format of the token_management_constraint_descriptor is specified in Table 48.

Table 48 – Format of token_management_constraint_descriptor

Field	Length	Type
token_management_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
token_constraint_type	2	uimsbf
token_unit_length	3	uimsbf
token_consumed_length	3	uimsbf
token_unit	8*token_unit_length	uimsbf
for(i=0;i<token_consumed_length;i++){		
token_consumed	8*token_consumed_length	uimsbf
}		
if (token_constraint_type==0x2) {		
timer	16	uimsbf
}		
}		

length – The number of bytes following this field.

token_constraint_type – If the value of this field equals 0x0 (COUNT) or 0x2 (TIMED_COUNT), the consumption of the DRM content shall be counted and any consumption of the DRM content equalling the number of "counts" as indicated by the token_unit field requires the consumption of the amount of tokens as indicated by the value of the token_consumed field.

If the value of this field equals 0x1 (DURATION), any consumption of the DRM content with a duration of the number of seconds as indicated by the token_unit field requires the consumption of the amount of tokens as indicated by the value of the token_consumed field.

All other values of this field are reserved for future use.

token_unit_length – Field defining the length in bytes of the token_unit field. The value shall not be bigger than 4.

token_consumed_length – Field defining the length in bytes of the token_consumed field. The value shall not be bigger than 4.

token_unit – If the token_constraint_type field equals 0x00 (COUNT) or 0x2 (TIMED_COUNT), the token_unit indicates the amount of "counts" of consumption of the DRM content that can be consumed for the amount of tokens as indicated in the token_consumed field.

If the token_constraint_type field equals 0x01 (DURATION), the token_unit indicates the number of seconds of consumption of the DRM content that can be consumed for the amount of tokens as indicated in the token_consumed field.

token_consumed – This field indicates the amount of tokens that shall be consumed from the token store of the device if the amount of DRM content is consumed as indicated by the token_constraint_type field and the token_unit field.

timer – Specifies the number of seconds after which the count state is reduced starting from beginning to render the content in the case that the value of the token_constraint_type field has the value 0x2 (TIMED_COUNT).

9 Registration layer

9.1 General

In this clause, the registration layer is specified. The registration layer is layer 4 of the four-layer model (see 5.4) used in the protection functionality specified by this standard. The specific use of the registration layer for the individual supported systems is further specified in 13.5, 14.5, 15.6 and 16.5.

9.2 RI context

In order to communicate with a rights issuer (a.k.a. RI) and obtain rights objects from the RI, the device needs to have an RI context for that right issuer. To obtain an RI context the device notifies its device data to the RI. The RI will then contact a root of trust (a.k.a. ROT) to request the certificate and capabilities matching this device data by checking the data against the public key infrastructure (PKI) and a certificate revocation list (CRL) from that root of trust. If the ROT certifies that the notified device data is valid, the RI can decide to send registration data to the device. The device will create an RI context from that registration data.

There are two types of RI context, being:

- RI context for interactive mode of operation. This is specified in OMA-ERP-DRM-V2_0.
- RI context for broadcast mode of operation. This is specified in 9.3.2.

NOTE Both types of RI context are different from each other.

In the case that the device supports an interactivity channel that can be used to contact an RI, the device is called an interactive device. In the case where the device does not have a direct interactivity channel to contact the RI, the device is called a broadcast device.

The sending of registration data to the device is handled by a registration operation. There are several different types of registration operations, which are defined in Table 49.

Table 49 – Registration types

Registration type	Device types	Subclause
Register device for interactive mode of operation	Interactive	9.3.1
Register device for broadcast mode of operation (content and RO via broadcast channel)	Broadcast and mixed-mode	9.3.2
Register device for mixed-mode of operation (both interactive and broadcast)	Mixed-mode	9.3.5

On successful execution, the registration operations result in the creation of an RI context in the device. An RI context for broadcast mode of operation will differ from an RI context for interactive mode of operation.

9.3 Registration layer protocols and message specification

9.3.1 Interactivity channel registration layer specification

Registration over an interactivity channel is according to OMA-ERP-DRM-V2_0, using the standard 4-Pass ROAP.

This encompasses devices with an interactivity channel, as well as unconnected devices, which have the capability to make a connection to an interactive device, as specified in OMA-ERP-DRM-V2_0, section 14, via the OBEX protocol and use the interactive device to report the device data to the RI.

9.3.2 Broadcast channel registration layer specification

9.3.2.1 Broadcast channel registration layer protocol overview

To register a device, the device data has to be notified to the RI. There are two cases for the notification of device data to the RI.

Case 1: The device has never been registered before and is activated by the user.

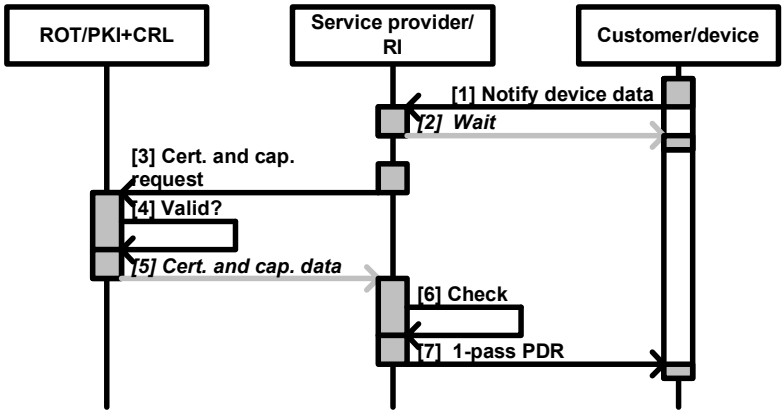
There are two possibilities in which the device has no direct communication back channel to contact the RI but needs to report device data to the RI.

- The device has no interactivity channel or an interactivity channel is not able to make a connection to the RI, but the device is able to create another connection to a connected OMA device. This device is called an unconnected interactive/unconnected mixed-mode device and is covered in 9.3.1.
- The device has no interactivity channel and is unable to make a connection to an interactive device. This device is called a broadcast device. In this case, the 1-pass binary push registered device protocol is used, as is specified in this standard.

Case 2: The device has been registered at the RI before and needs to be re-registered.

- In this case the RI uses the 1-pass binary inform registered device protocol to send a message ordering the device to re-register, as is specified in this standard.

The following sequence chart (see Figure 22) explains the registration for broadcast only mode of operation.



NOTE Notification of device data to the rights issuer is performed off-line. Transmission of the registration data from the RI to the device is performed on line via a broadcast channel.

Figure 22 – Registration for broadcast mode of operation with one ROT

Explanation of the protocol:

- Once the RI has the device data from the device (1) via the protocol specified in 9.3.2.2, the RI contacts the ROT (3), while the device is entered into registration mode and awaits the registration data (2).
- The ROT implements a public key infrastructure (a.k.a. PKI). The PKI looks up the certificate and capabilities belonging to the device data in question (4). The ROT should have a certificate revocation list (CRL). In any case, it is the responsibility of the ROT to decide whether the requested device data is valid or not and whether or not the requested certificate and capabilities data can be passed to the RI.
- Assuming the RI received the requested certificate and capabilities from the ROT (5), the RI will perform some last checks (6) and shall send back a registration data message to the device (7).
- The RI uses the 1-pass binary push device registration data (a.k.a. PDR) protocol to send the registration data over the broadcast network. The PDR protocol is specified in 9.3.2.4. The registration data (in the format of the device_registration_response() message) is specified in 9.3.2.7.2. The RI may decide to send an error status with the message or send valid registration data containing the data required to create an RI context.
- A device listening for device_registration_response() messages will look for messages with the corresponding message_tag. On every message with a matching message_tag, the device will check the long_form_udn parameter. If this matches (any of) the device's local UDN(s) the device will process the message and will start trying to decrypt the secret data in it.
- If the device does not receive registration data within a timeout, the device leaves the registration mode and stops listening for device_registration_response() messages.

After successful execution of the above (re-) registration protocol (that means that a valid RI context is present), a device is able to receive and use BCROs from that RI for the consumption of content. BCROs are distributed using the RI services (Clause 11).

The above results in the specification of several protocols:

- offline protocols (from device to RI)

Protocol	Subclause	Purpose
Off-line notification of detailed devicedata protocol	9.3.2.2	Registration
Off-line notification of short devicedata protocol	9.3.2.3	Inform RI by action request codes

- 1-pass protocols (from RI to device)

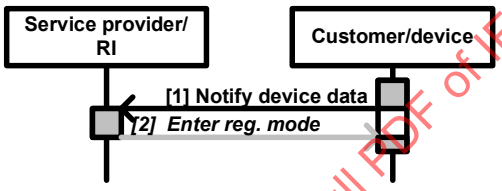
Protocol	Subclause	Purpose
1-pass binary push device registration protocol	9.3.2.4	Transmit registration data to device
1-pass binary inform registered device protocol	9.3.2.5	Inform device via messages

- supporting protocols for registration

Protocol	Subclause	Purpose
Unique device number (UDN) protocol	9.3.2.6	Make registration data robust (part of off-line notification of detailed device data)

9.3.2.2 Offline notification of detailed devicedata protocol

This protocol is also known as the "off-line NDD protocol", short for off-line notification of detailed data protocol.



NOTE Notification of device data is performed off-line. The device data (device_data_inform() message) is defined in 9.3.2.7.

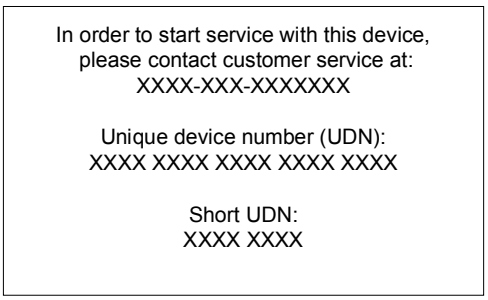
Figure 23 – Offline NDD protocol

Explanation of the protocol (see also Figure 23).

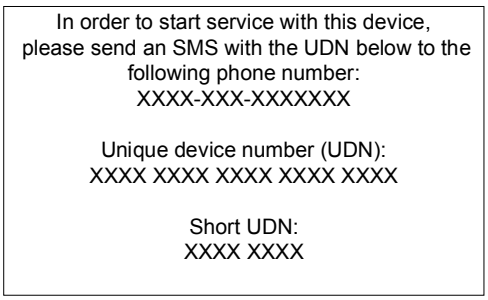
- The device shall notify (1) its device data via some means to the RI. After user interaction, the device shall produce the device_data_inform() message (see 9.3.2.7.1 for details) and make this data available to the user.
- The device may display a dialogue with instructions. Notifying the device data can be done in various ways, for example, by showing the user of the device a dialogue on the screen of the device, displaying the device data and a telephone number to call for vocal notification of the device data. Another example is to display instructions to send an SMS message via a mobile phone to the RI.

An example of a displayed message follows, where the following information is reported back to the RI:

NOTE It is the sequence of the defined values that is specified. The use of dashes as the delimiter is shown with an example placement to be consistent with the examples used elsewhere in this standard. The text portion of this screen is shown as an example only (see Figure 24); there is no implied requirement to duplicate the exact wording or formatting shown. The numeric fields should be included as defined above. The short UDN will only be displayed after the first registration, when that data may be available for display.



An example dialogue showing instructions for vocal notification of UDN to the call centre.



An example dialogue showing instructions for notification of UDN by SMS to the call centre.

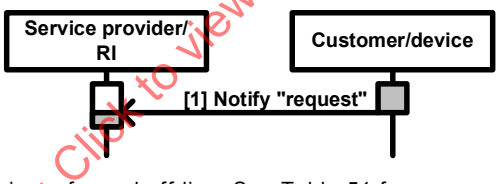
Figure 24 – Samples of notification displays

- If the device does not support a return channel to the RI, the device data (device_data_inform() message) shall be notified off-line, using the off-line notification of detailed devicedata protocol. The device data to notify shall be reduced by a special protocol (see 9.3.2.6).
- After the notification of the device data, user needs to put the device into registration mode (2). When put into registration mode, the device shall start to listen for the device registration data for a limited time.

9.3.2.3 Offline notification of short devicedata protocol

9.3.2.3.1 General

This protocol is also known as the "off-line NSD protocol", short for off-line notification of short data protocol.



NOTE Notification of device data is performed off line. See Table 51 for an overview of the possible "requests".

Figure 25 – Off-line NSD protocol

Explanation of the protocol (see also Figure 25):

- The user may notify a short decimal code called the action request code (ARC) to the RI via offline methods (for example, telephone call or SMS or other). The code shall be constructed as indicated in Figure 26 and Table 50.

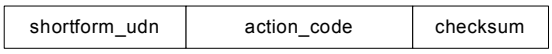


Figure 26 – Action request code (ARC)

NOTE 1 For some of the ARCs (for example, the ARC token_consumption_report), the user may have to notify more digits to the RI than the ones of the ARC.

Table 50 – NSD action request code fields

ARC fields	Length (digits)	Supporting up to
shortform_udn	8	100 million devices
action_code	2	99 action codes
checksum	2	

This totals 12 digits. The fields are defined below.

shortform_udn – The off-line notification can be performed faster if the long-form UDN is not used, but a shorter form instead. After first-time notification of the device data to the RI, the RI may issue a short version of the full UDN (called shortform_udn) that is carried in the device_registration_response() message. The shortform_udn number is used to speed up the off-line interaction with the RI. If this number is stored into the device, subsequent "requests" by the user of the device can be notified off line much quicker by using the shortform_udn number concatenated by a standardized action code.

NOTE 2 In cases where the device needs to be identified uniquely in another network than its home network where it was registered, the shortform_udn cannot be used because the (new/different) RI does not have the shortform_udn in its database. In this case, the only possibility for the hosting RI to identify the device uniquely would be via the longform_udn. It is the responsibility of the device to decide when it is appropriate to use the longform_udn instead, for example, by comparing the service operations centre (SOC) ID received with the SOC ID remembered from registration.

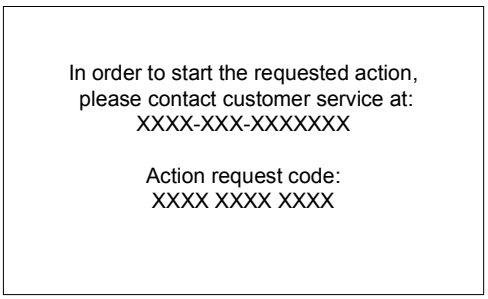
action_code – Following the shortform_udn, the user of the device can notify an action code to the RI. The NSD protocol defined in this standard shall use action_codes from Table 51 to construct the ARC.

Table 51 – NSD action types

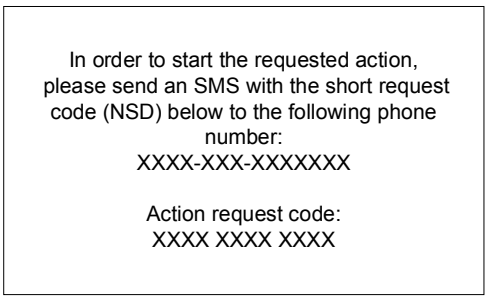
Action type	Action code (d)	Specified in subclause
Re-registration (only at same RI)	{0d01}	9.3.2.3.2
Resend BCRO	{0d02}	13.7.8
Reserved for future use	{0d03,..,0d09}	
Join domain	{0d10,..,0d19}	9.3.2.3.3
Leave domain	{0d20,..,0d29}	9.3.2.3.4
Purchase	{0d30}, whereas content identification is supplied by the Service guide.	12.4.2
token_consumption_report	{0d31,..,0d39}	9.3.2.3.5
Reserved for future use	{0d40,..,0d49}	
token_request	{0d50,..,0d59}	9.3.2.3.6
Reserved for future use	{0d60,..,0d69}	
Notify DRM time drift	{0d70,..,0d89}	9.3.2.3.7
TAA report	(0d90}	9.3.2.3.8
Reserved for future use	{0d91,..,0d99}	

checksum – The constructed shortform_udn and action_code is appended by checksum digits. See Clause A.9 for an explanation of the algorithm.

An example: In order to request to re-register, a sample NSD action request code could look like: "1660 8731 0112". An example of a displayed message is shown in Figure 27, where the information to be reported back to the RI is shown.



An example dialogue showing instructions for vocal notification of ARC to the call centre.



An example dialogue showing instructions for notification of ARC by SMS to the call centre.

It is the sequence of the defined values that is specified. The use of dashes as the delimiter is shown with an example placement to be consistent with the examples used elsewhere in this standard. The text portion of this screen is shown as an example only; there is no implied requirement to duplicate the exact wording or formatting shown. The numeric fields shall be included as defined above. The short UDN will only be displayed after the first registration, when that data may be available for display.

Figure 27 – Samples of notification displays showing an ARC message

9.3.2.3.2 Request re-registration (only at same RI)

After sending this ARC, the user will wait until he receives the confirmation of the RI in the form of a device_registration_response() message (see 9.3.2.4).

9.3.2.3.3 Request join domain

The action request code (ARC) for the NSD protocol is formed according to following rules:

- the first digit is used to notify the join domain action;
- the second digit is used as a device_nonce to help the device to keep track of join domain requests.

After notifying the ARC to the RI, the user may notify a particular domain group number identifying a domain where the device is to be entered. The RI shall incorporate the device_nonce from the request in the response message.

9.3.2.3.4 Request leave domain

The action request code (ARC) for the NSD protocol is formed according to following rules:

- the first digit is used to notify the leave domain action;
- the second digit is used as a device_nonce to help the device to keep track of leave domain requests.

After notifying the ARC to the RI, the user needs to notify a particular domain group number identifying a domain where the device is to be removed from. The device shall display a domain ID. The RI shall incorporate the device_nonce from the request in the response message.

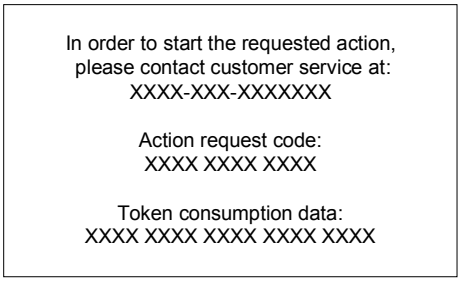
9.3.2.3.5 Token consumption report

The action request code (ARC) for the NSD protocol is formed according to following rules:

- the first digit is used to notify the token consumption report;
- the second digit is used as a device_nonce to help the device to keep track of token consumption reports.

After notifying the ARC to the RI, the user should notify the token consumption data. The device shall display the token consumption data, for example, to the left of or below the digits of the ARC for the token consumption report. The RI shall incorporate the device_nonce from the request in the response message.

An example of a displayed message is shown in Figure 28, where the information to be reported back to the RI is shown.



It is the sequence of the defined values that is specified. The use of dashes as the delimiter is shown with an example placement to be consistent with the examples used elsewhere in this standard. The text portion of this screen is shown as an example only; there is no implied requirement to duplicate the exact wording or formatting shown. The numeric fields shall be included as defined above. The short UDN will only be displayed after the first registration, when that data may available for display.

Figure 28 – Sample of token consumption reporting notification display

The token consumption data are defined in Table 52.

Table 52 – Token consumption data

Field	Length (digits)	Supporting up to
tokens_consumed	4	9999 tokens to be reported
report_authentication_code	13	
checksum	3	

This totals 20 digits. The fields are defined below.

tokens_consumed – This field contains the amount of tokens the device wished to report as consumed to the RI. See Clause A.16 for more information.

report_authentication_code – This field contains the authentication code for the value in the tokens_consumed field and the value of the device_nonce (second digit of the action_code of the ARC of this message). See Clause A.15 for the computation of the report authentication code.

checksum – The final digits of the device ID number are check digits, akin to a checksum. The three digits allow 1 out of 10^3 possible errors to remain undetected. The checksum algorithm used is the UDN checksum; see A.9.2.

9.3.2.3.6 Token request

The action request code (ARC) for the NSD protocol is formed according to following rules:

- the first digit is used to notify the token request;
- the second digit is used as a device_nonce to help the device to keep track of token requests.

After notifying the ARC to the RI, the user should notify the number of tokens desired and the RI may request additional data (such as a bookable account). The RI shall incorporate the device_nonce from the request in the token_delivery_response message.

9.3.2.3.7 Notify DRM time drift

Time drift is expressed in minutes and rounded up to next multiple of 5 min. The range is 0..90 min, whereas value 89 will decode as timedrift > 90 min. Some examples of valid ARC values are given below.

EXAMPLE 1 Device notifies 2 min timedrift from newly received DRM time message: action code is 71.

EXAMPLE 2 Device notifies 38 min timedrift from newly received DRM time message: action code is 78.

EXAMPLE 3 Device notifies 235 min timedrift from newly received DRM time message: action code is 89.

9.3.2.3.8 TAA report

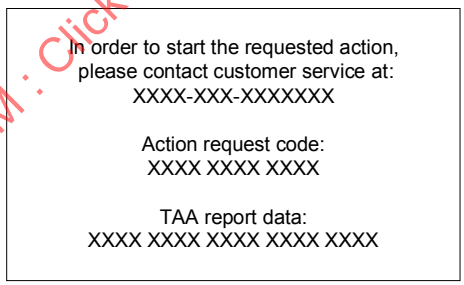
The TAA report may be used to check whether a device has implemented the TAA algorithm (see A.11.7).

The action request code (ARC) for the NSD protocol is formed according to following rules:

- the first digit is used to notify the TAA report;
- the second digit is 0.

After notifying the ARC to the RI, the user should notify the TAA report data. The device shall display the TAA report data, for example, to the left of, or below, the digits of the ARC for the TAA report.

An example of a displayed message is shown in Figure 29, where the information to be reported back to the RI is shown.



It is the sequence of the defined values that is specified. The use of dashes as the delimiter is shown with an example placement to be consistent with the examples used elsewhere in this standard. The text portion of this screen is shown as an example only; there is no implied requirement to duplicate the exact wording or formatting shown. The numeric fields shall be included as defined above. The short UDN will only be displayed after the first registration, when that data may available for display.

Figure 29 – Sample of TAA report display

The TAA report data are defined in Table 53.

Table 53 – TAA report data

Field	Length (digits)
TAA_report_code	17
checksum	3

This totals 20 digits. The fields are defined below.

TAA_report_code – This field contains the TAA_report_code. See A.8.7 for the computation of the TAA_report_code and when it shall be computed and presented to the user.

checksum – The final digits of the device ID number are check digits, akin to a checksum. The three digits allow 1 out of 10^3 possible errors to remain undetected. The checksum algorithm used is the UDN checksum, see A.9.2.

9.3.2.4 1-pass binary push device registration protocol

This protocol is also known as the "1-pass PDR protocol", short for push device registration protocol.

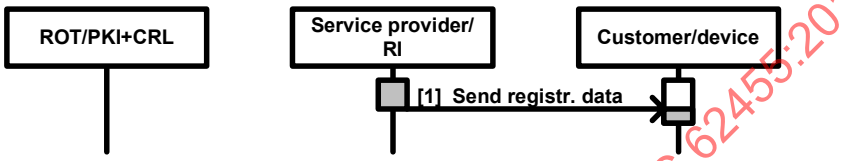


Figure 30 – 1-pass PDR protocol – (first) device registration

The transmission of registration data is performed on-line via a broadcast channel. The registration data (device_registration_response() message) is specified in 9.3.2.7.2.

Explanation of the protocol (see also Figure 30):

- The RI shall use the 1-pass binary push device registration data (a.k.a. PDR) protocol to send registration data over the network (1). The registration data can be the device_registration_response() message (see 9.3.2.7.2) or the domain_registration_response() message (see 9.3.3.5.1). The RI shall use the mechanisms specified in 13.7.5 to address the message to a device. The RI shall include a valid keyset in the message.
- A device listening for device_registration_response() (or domain_registration_response()) messages shall look for messages with the corresponding message_tag. On every message with a matching message_tag, the device shall check the long_form_udn parameter. If this matches (any of) the device's local UDN(s) the device shall start validating the signature and check the RI certificate (chain). If both (UDN and signature) are valid, the device detects this message is really addressed to it. The device shall start processing the message and shall start trying to decrypt the secret data in it. If the message is correct, the device shall store the new keyset with key(s). The device shall delete the old keyset (if applicable).
- After a timeout the device shall leave the registration mode and stops listening for device_registration_response() messages.

9.3.2.5 1-pass binary inform registered device protocol

9.3.2.5.1 General

This protocol is also known as the "1-pass IRD protocol", short for inform registered device protocol.

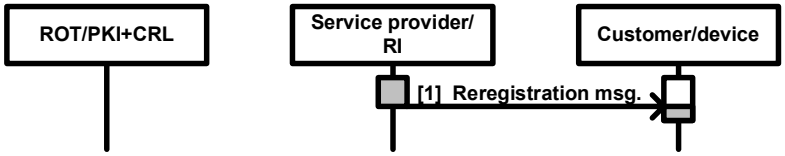


Figure 31 – 1-pass IRD protocol – RI initiated message to device (here re-registration)

Explanation of the protocol (see also Figure 31 and Table 54):

- The 1-pass IRD protocol is designed to meet the messaging push case. Its successful execution assumes the device to have an existing RI context with the sending RI.
- Several messages are defined for the IRD protocol.

Table 54 – Messages of the 1-pass IRD protocol

Message name	For message syntax, see subclause	Remark
Force to re-register	9.3.2.7.3	
Update RI certificate	9.3.2.7.4	
Update DRM Time	9.3.2.7.5	
Update contact number	9.3.2.7.6	
Present TAA report	9.3.2.7.7	
Update domain	9.3.3.5.2	
Force to join domain	9.3.3.5.3	
Force to leave domain	9.3.3.5.4	
Token delivery	9.3.4.4	

The processing of each message will be specified in the following subclauses.

9.3.2.5.2 Force re-registration

In this case, the RI is sending a message to the device to get it into registration mode.

- The RI shall use the mechanisms specified in 13.7.5 to address the message to a device.
- The device shall filter on the message_tag to identify the message. Then it shall filter for the UDN and compare it to the local UDN of the device. If those match, the device shall start validating the signature and check the RI certificate (chain). If both (UDN and signature) are valid, the device detects this message is really addressed to it, and the device shall start to perform the intended action.
- If the message is correct, the reception of this message shall start the (re-)registration process. The device will be rendered inoperable but only in relation with the associated RI (context) as specified below.
  - Accessing a service guide for purchase is still allowed, as this will require a registration first.
  - The device shall be rendered inoperable for any purchase protocol or playback of future content. The device may use stored BCROs to play old content for which the device obtained ROs, but shall not use these BCROs for new content received after the re-registration request until the device is re-registered with the RI.
- Depending on the implementation, a dialogue will be shown to the user and the off-line NDD protocol will be executed, using the RI_ID stored in the RI context. Depending on the implementation, a dialogue may be shown to the user and the off-line NDD protocol shall be executed.

9.3.2.5.3 Update RI certificate

The RI can use this message to update the RI certificate in one or more devices.

- The RI shall enter a valid RI certificate in the message.
- The RI may enter a rooted RI certificate chain in the message. The root certificate is to be excluded.
- The RI shall use the mechanisms specified in 13.7.5 to address the message to a device.
- The device shall filter on the message_tag to identify the message. Then it shall filter for the UDN and compare it to the local UDN of the device. If those match, the device shall

start validating the signature and check the RI certificate (chain). If both are valid, the device detects this message is really addressed to it, and the device shall start to perform the intended action.

- If the message is correct, the device shall save the new RI certificate in the message after the signature of the message has been verified correctly. The old RI certificate shall be made obsolete.

9.3.2.5.4 Update DRM_Time

The RI can use this message to update the DRM time.

- The RI shall enter a valid DRM time in the message.
- The RI may put a time offset in the message. The time offset shall be valid.
- The RI shall use the mechanisms specified in 13.7.5 to address the message to a device.
- The device shall filter on the message_tag to identify the message. Then the device shall start validating the signature and check the RI certificate (chain). If both are valid the device detects this message is really addressed to it, and the device shall start to perform the intended action.
- If the message successfully validated and the RI certificate is valid, the device shall save the new DRM time into the device.

9.3.2.5.5 Update contact number

The RI can use this message to update the contact number that the device should contact during the offline notification processes (both for use with the NDD or NSD protocols).

- The message shall contain (a) valid telephone number(s) to contact.
- The RI shall use the mechanisms specified in 13.7.5 to address the message to a device.
- The device shall filter on the message_tag to identify the message. Then the device shall start validating the signature and check the RI certificate (chain). If both are valid the device detects this message is really addressed to it and the device shall start to perform the intended action.
- If the message is correct, the device shall store the new contact number(s) and delete the old one(s).

9.3.2.5.6 Force to join a domain

In this case, the RI is sending a message to the device to get it into join domain mode, which may be followed up by the matching action in the NSD protocol.

- The RI shall use the mechanisms specified in 13.7.5 to address the message to a device.
- A device listening for device_registration_response() messages shall look for messages with the corresponding message_tag. On every message with a matching message_tag, the device shall check the long_form_udn parameter. If this matches (any of) the devices local UDN(s) the device shall start validating the signature and check the RI certificate (chain). If both (UDN and signature) are valid the device detects this message is really addressed to it, and the device shall start to perform the intended action.

9.3.2.5.7 Force to leave a domain

In this case, the RI is sending a message to the device to get it into leave domain mode, which may be followed up by the matching action in the NSD protocol.

- The RI shall use the mechanisms specified in 13.7.5 to address the message to a device.
- A device listening for device_registration_response() messages shall look for messages with the corresponding message_tag. On every message with a matching message_tag, the device shall check the long_form_udn parameter. If this matches (any of) the devices local UDN(s) the device shall start validating the signature and check the RI certificate

(chain). If both (UDN and signature) are valid the device detects this message is really addressed to it and the device shall start to perform the intended action.

9.3.2.5.8 Update a domain

The RI can use this message to inform the device that it left a particular domain.

- The message shall contain a valid domain id.
- The RI shall use the mechanisms specified in 13.7.5 to address the message to a device.
- A device listening for device_registration_response() messages shall look for messages with the corresponding message_tag. On every message with a matching message_tag, the device shall check the long_form_udn parameter. If this matches (any of) the device's local UDN(s) the device shall start validating the signature and check the RI certificate (chain). If both (UDN and signature) are valid, the device detects this message is really addressed to it and the device shall start to perform the intended action.
- If the message is correct, the device shall delete the associated domain context.

9.3.2.6 Unique device number (UDN) protocol

9.3.2.6.1 Protocol specification

To reduce the amount of data that is to be notified to the RI, the device data protocol takes care of data reduction. To ease the detection of errors during the registration process, the device data protocol will also allow detection of errors in the notified device data.

The algorithm shown in Figure 32 and Table 55 shall be used to construct a unique device number (a.k.a. UDN).

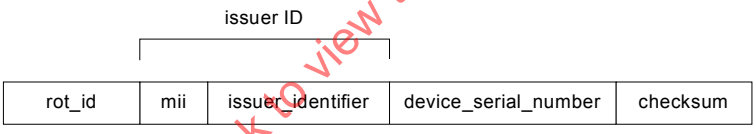


Figure 32 – Unique device number

Table 55 – UDN explanation

Field	Length (digits)	Supporting up to
rot_id	3	1 000 ROT
mii	1	9 major industries
issuer_identifier	4	100 000 issuers (10 000 in 10 industries)
device_serial_number	9	1 billion
checksum	3	

This totals 20 digits. The fields are specified below.

Every of 1 000 ROT can issue 100 000 issuer ranges, from which every unique issuer can have 1 billion devices issued.

rot_id – The first three digits in the UDN identify the ROT. Every ROT has an own unique ID.

mii – The first digit of the device ID number is the major industry identifier (MII), which represents the category of entity, which issued the device_serial_number. Different MII digits represent the issuer categories shown in Table 56.

Table 56 – Major industry identifier

MII digit value	Issuer category	Remarks
0	Root of trust	
1	Telecom	
2	Consumer electronics	
3	Network equipment	
4	Reserved for future use	
5	Reserved for future use	
6	Reserved for future use	
7	Reserved for future use	
8	Reserved for future use	
9	National assignment	
EXAMPLE Philips is in the consumer electronics category and Nokia is in the Telecom sector. If the MII digit is 9, then the next three digits of the issuer identifier are the three-digit country codes defined in ISO 3166, and the remaining final two digits of the issuer identifier can be defined by the national standards body of the specified country in whatever way it wishes.		

issuer_identifier – The issuer_identifier identifies which issuer created the devices serial number. Together with the MII digit, this forms the issuer ID.

device_serial_number – The device_serial_number is unique inside a range of an issuer. Each issuer therefore has 1 billion (10^9) possible device_serial_numbers. It is unlikely that an issuer exceeds 1 billion serial numbers. An issuer wishing to group devices in another way can request a second issuer_identifier for another range (of 1 billion).

checksum – The final digits of the device ID number are check digits, akin to a checksum. The three digits allow 1 out of 10^3 possible errors to remain undetected. See Clause A.9 for an explanation of the algorithm.

9.3.2.6.2 Message syntax

The 20 digits of the UDN are encoded in BCD format into the longform_udn(). The message syntax is specified in Table 57.

Table 57 – longform_udn

Field	Length	Type
longform_udn(){		
rot_id	12	bslbf
mii	4	bslbf
issuer_identifier	16	bslbf
device_serial_number	36	bslbf
checksum	12	bslbf
}		

9.3.2.7 Binary messages

9.3.2.7.1 Device data – device_data_inform() message

9.3.2.7.1.1 Message specification

The device data shall prove that it is unique. In a one-way case, the device notifies this device data, yet the length of the unique device data should remain concise.

Because devices can be uniquely identified by the PKI, it is not needed to incorporate unique data like the device certificate into the (device-specific) registration data. The OMA DRM 2.0 certificate is global, and the link between the manufacturer and the device can be requested from the PKI, based on the device ID.

The parameters of the device_data_inform() message are specified in Table 58.

Table 58 – Notify device data message parameters

Device_Data_Inform()		
parameter	(M)andatory / (O)ptional[a]	Remark
version	M	
contact_number	O	
longform_udn()	M	

[a] (O)ptional means that the user of the message may include the parameter in the message, but the device shall support the interpretation of the parameter.
(M)andatory means that the user of the message shall include the parameter in the message.

version – Is a <major> representation of the highest ROAP version number supported by the device. For this version of the protocol, the *version* field shall be set to value '1'.

contact_number – Is the number to be contacted in order to register the device. It can be a phone number or an SMS number. This number may have been entered into the device at production time and if so may be shown in the registration display (see 9.3.2.2 for an example). This number could also be provided in other ways, like a leaflet in the package of the device, a commercial channel that is viewed after selection of a free to air channel via the service guide or via entirely other means.

longform_udn() – Identifies the unique_device_number to the RI. The UDN shall be part of the credentials entered at production time into the device, like the private key and the certificate. See 9.3.2.6 for details.

9.3.2.7.1.2 Message syntax

Since this is an offline protocol, the device data is not really formed into a message that can be transmitted. The device data is decimal and formatted as specified in Table 59.

Table 59 – Device data

Parameter	Format and length	Description
version	1 byte	
contact_number	15 bytes	Dependent on target telco network
longform_udn()	20 bytes	UDN protocol

9.3.2.7.2 Registration data – device_registration_response() message

9.3.2.7.2.1 Message specification

Using the 1-pass PDR protocol the RI shall send a device_registration_response() message with the registration data to the device as specified in Table 60.

Table 60 – Message fields

Device_Registration_Response()		
Parameter name	**(M)andatory/(O)ptional[a]**	**Remark**
message_tag	M	Global, not encrypted
protocol_version	M	Global, not encrypted
longform_udn ()	M	Global, not encrypted
status	M	Device specific, not encrypted
certificate_version	M	Global, not encrypted
ri_certificate_counter	M	Global, not encrypted
c_length	M	Global, not encrypted
ri_certificate	M	Global, not encrypted
ocsp_response_counter	M	Global, not encrypted
r_length	M	Global, not encrypted
ocsp_response	M	Global, not encrypted
local_time_offset_flag	M	Device specific, not encrypted
time_stamp_flag	M	Device specific, not encrypted
subscriber_group_key_flag	M	Device specific, not encrypted
signature_type_flag	M	Global, not encrypted
shortform_udn_flag	M	Device specific, not encrypted
surplus_block_flag	M	Device specific, not encrypted
keyset_block_length	M	Device specific, not encrypted
unique_group_key	O	Device specific, encrypted
subscriber_group_key	O	Device specific, encrypted
unique_device_key	O	Device specific, encrypted
unique_device_filter	M	Device specific, encrypted
customer_group_filter	O	Device specific, encrypted
ri_authentication_key	M	Device specific, encrypted
token_delivery_key	O	Device specific, encrypted
TAA_descriptor	O	Device specific, encrypted
broadcast_domain_key	O	Device specific, encrypted
shortform_domain_id	O	Device specific, encrypted
drm_time	M	Device specific, not encrypted
local_time_offset	O	Device specific, not encrypted
registration_timestamp_start	O	Device specific, not encrypted
registration_timestamp_end	O	Device specific, not encrypted
shortform_udn	O	Device specific, not encrypted
signature_block	M	Device specific, not encrypted

[a] (O)ptional means that the user of the message may include the parameter in the message, but the device shall support the interpretation of the parameter.
(M)andatory means that the user of the message shall include the parameter in the message.

message_tag – This parameter identifies the type of the message. See Clause A.12 for the value of the message_tag.

protocol_version – This parameter indicates the protocol version of this message. The device shall ignore messages that have a protocol_version number it does not support. The value of the protocol_version of this message shall be set to 0x0 (i.e., the original format).

If this parameter is set to 0x0, the format specified in this standard is used. If set to anything else than 0x0, the format is beyond the scope of this version of this standard.

longform_udn() – The long form of the UDN. See 9.3.2.6 for details.

status – The status parameter shall indicate one of the values defined in Table 61. The device shall ignore messages with other error values.

Table 61 – Status values

Status value	Meaning
Success	The registration request was executed successfully and the RI completed all data. The device shall process the message.
UnknownError	The RI encountered an unknown error after receiving the registration request. The device may put forward a subsequent registration request to the RI (context).
NotSupported	The RI does not support the registration request.
AccessDenied	The RI decided that the device will not be granted access to the service and stops the registration. The RI will stop listening to future registration requests of this device. The device is forced to refrain from future registration and shall **suppress** broadcast and/or mixed-mode registration requests to the particular RI (context).
NotFound	The RI decided that the device could not be found (off-line UDN and/or UaProf). The device may put forward a subsequent registration request to the RI (context).
MalformedRequest	The RI decided that the registration request was malformed and will **force** the device to execute a (re)-registration at once. The device shall enter (re)registration mode (see 9.3.2.5.2)

See Clause A.2 for the value of the error codes.

certificate_version – This parameter is a numerical representation of the version of the RI certificate; see Table 62. Using the certificate_version parameter the device can decide if it is needed to update the RI certificate (if it was stored before).

Table 62 – Fields of certificate_version parameter

Parameter fieldname	Field value ($_h$)	Supports
major_version_number	0x0,..,0xF	MSB4(certificate_version)
minor_version_number	0x0,..,0xF	LSB4(certificate_version)

The parameter is divided into two fields of four bits, whereas the first four bits (msb left) express the major number and the last four bits (lsb right) express the minor version. The major and minor number encode in bslbf format. Sixteen major and 16 minor versions are supported.

EXAMPLE Major.Minor version <1.2> is expressed as 0001 0010_b.

ri_certificate_counter – This parameter indicates the depth of the RI certificate chain; see Table 63.

Table 63 – Allowed values for ri_certificate_counter

Number of certificates in chain	Value (h)	Remark
0	0x0	Will signal absence of ri_certificate, for example, on error status to save bandwidth
1	0x1	
2	0x2	
3	0x3	
4	0x4	
5	0x5	
6	0x6	
7	0x7	
NOTE The certificate chain can have a depth of up to seven RI certificates.		

c_length – This parameter indicates the length in bytes of the ri_certificate.

ri_certificate() – This parameter shall be present. When present, the value of a *ri_certificate* parameter shall be a certificate chain including the RI's certificate. The chain shall not include the root certificate. The RI certificate shall come first in the list. Each following certificate shall directly certify the one preceding it.

The device may store RI certificate verification data indicating that an RI certificate chain has been verified. The purpose of this is to avoid repeated verification of the same certificate chain. The RI certificate verification data stored in this way shall uniquely identify the RI certificate and shall be integrity protected. The device should check if the RI certificate chain received in this parameter corresponds to the stored certificate verification data for this RI. If so, the device need not verify the RI certificate chain again; otherwise, the device shall verify the RI certificate chain.

If an RI certificate is received that is not in the stored certificate verification data for this RI, and if the device can determine (in the case of broadcast devices that support DRM time) that the expiry time of the received RI certificate is later than the RI context for this RI, and the certificate status of the RI certificate as indicated in the OCSP response is good (see OCSP-MP), then the device shall verify the complete chain and should replace the stored RI certificate verification data with the received RI certificate data and set the RI context expiry time to that of the received RI certificate expiry time.

However, if the device does store RI certificate verification data in this way it shall store the expiry period of the RI's certificate (as indicated by the notAfter field within the certificate) and shall compare the device's current DRM time with the stored RI certificate expiry time whenever verifying the signature on signed messages from the RI. If the device's current DRM time is after the stored RI certificate expiry time then the device shall abandon processing the RI message and shall initiate the registration protocol.

ocsp_response_counter – This parameter indicates the depth of the OCSP response chain, see Table 64.

Table 64 – Allowed values for ocsp_response_counter

Number of responses in chain	Value ($_h$)	Remark
0	0x0	Will signal absence of ocsp_response, for example, on error status to save bandwidth
1	0x1	
2	0x2	
3	0x3	
4	0x4	
5	0x5	
6	0x6	
7	0x7	
NOTE The certificate chain can have a depth of up to seven OCSP responses.		

r_length – This parameter indicates the length in bytes of the ocsp_response.

ocsp_response() – This parameter, when present, shall be a complete set of valid OCSP responses for the RI's certificate chain. The device shall not fail due to the presence of more than one OCSP response element. A device shall check that an OCSP response is present in the received message. If no OCSP response is present in the device_registration_response() message, then the device shall abort the registration protocol.

local_time_offset_flag – Binary flag to signal the presence of the local_time_offset field. See Table 65 for the allowed values and their meaning.

Table 65 – Values for flags signalling data absent/data present

Value ($_h$)	Meaning
0x0	data absent
0x1	data present

time_stamp_flag – Binary flag to signal the presence of the registration_timestamp_start and registration_timestamp_end field. See Table 65 for the allowed values and their meaning.

subscriber_group_key_flag – The flag expresses how many subscriber_group_keys (a.k.a. SGK) are delivered with the registration data. When zero message broadcast is used, a set of 8 keys will support a group size of 256. A set of nine keys will support a group size of 512. Other values or larger group sizes are not supported. A value larger than zero indicates that the registration data message delivers a set of zero message subscriber_group_key (s) to the device and that the device needs to use zero message broadcast style encryption to deduce the decryption key to decrypt the SEK. See also Table 66.

Table 66 – Allowed values for subscriber_group_key_flag

Subscriber_group_key_flag	Value ($_h$)	Remark
Data absent	0x0	Will signal absence of keyset_block, for example, on error status to save bandwidth
Reserved for future use	0x1-0x7	Not used in this version of this standard
Set of (8) subscriber_group_key	0x8	
Set of (9) subscriber_group_key	0x9	
Reserved for future use	0xA-0xF	Not used in this version of this standard

signature_type_flag – A flag to signal the type of signature algorithm used. See Table 67 for the allowed values and their meaning.

Table 67 – Values and their meaning for signature_type_flag

Value ($_h$)	Meaning	Remark
0x0	RSA 1024	
0x1	RSA 2048	
0x2	RSA 4096	
0x3	Reserved for future use	Not used in this version of this standard

shortform_udn_flag – Binary flag to signal the presence of the shortform_udn field. See Table 65 for the allowed values and their meaning.

surplus_block_flag – Binary flag to signal the presence of the surplus_block() structure. See Table 65 for the allowed values and their meaning.

keyset_block_length – This parameter indicates the length in bits of the total keyset_block. That is the part in the sessionkey_block() plus the optional second part from the surplus_block().

unique_group_key – A symmetric AES encryption key to address a unique group. This key is also known as UGK. The key length shall be 128 bit.

NOTE 1 This key is wrapped into the keyset_block, see 9.3.2.7.2.2.

subscriber_group_key – An (set of) AES symmetric encryption key(s) which are used for the zero message subscriber_group_key deduction of the key needed to decrypt the SEK and/or PEK. These subscriber_group_key is also known as SGK. The key length shall be 128 bits.

NOTE 2 This key is wrapped into the keyset_block, see 9.3.2.7.2.2.

unique_device_key – An AES symmetric key to address a unique device. This key is also known as UDK. The key length shall be 128 bits.

NOTE 3 This key is wrapped into the keyset_block, see 9.3.2.7.2.2.

unique_device_filter – An EN 50094:1992 style addressing scheme used to filter for messages like BCROs. A device address consists of five bytes and is unique within an operation. In the case of a group size of 256 devices, the first 32 bits contain the group_address field, whilst the last 8 bits contain the position_in_group field. In the case of 512 devices, the first 31 bits contain the group_address field whilst the last 9 bits contain the position_in_group field. This address is also known as UDF.

NOTE 4 This address is wrapped into the keyset_block, see 9.3.2.7.2.2.

customer_group_filter – Device-specific data for filtering KSM messages using the black-out-spot-beam mechanism; see 7.2.2.4. The customer_group_filter is also known as CGF. During registration, the RI may define a CGF for the device to be part of the registration data in the keyset. If there is a blackout_spotbeam access_criteria_descriptor in a KSM, the length of its bosb_mask field shall be the same as the length of the CGF. A '1' on a certain position of the CGF indicates that the device belongs to a certain category. It is out of scope of this standard to define the categories. However, see Clause B.9 for examples of categories and their use.

NOTE 5 This address is wrapped into the keyset_block, see 9.3.2.7.2.2.

ri_authentication_key – An AES symmetric key to verify MACs on BCRO and KSM messages. This key is also known as RIAK. The key length shall be 128 bits.

NOTE 6 This key is wrapped into the keyset_block, see 9.3.2.7.2.2.

token_delivery_key – This is the token delivery key (TDK), which is used in 9.3.4.4.

NOTE 7 This descriptor is wrapped into the keyset_block, see 9.3.2.7.2.2.

TAA_descriptor – This is the TAA descriptor as specified in A.11.7.

NOTE 8 This descriptor is wrapped into the keyset_block, see 9.3.2.7.2.2.

broadcast_domain_key – An AES symmetric key to address a broadcast domain. This key is also known as BDK. The key length shall be 128 bits.

NOTE 9 This key is wrapped into the keyset_block, see 9.3.2.7.2.2.

longform_domain_id() – This parameter is also known as the long-form broadcast domain filter (LBDF). See A.11.6 for the definition. The longform_domain_id() is used for mixed-mode operation.

NOTE 10 This address is wrapped into the keyset_block, see 9.3.2.7.2.2.

shortform_domain_id – This parameter is also known as the short-form broadcast domain filter (SBDF); see A.11.1. An addressing scheme used to filter for messages like BCROs. The shortform_domain_id is used for broadcast mode of operation.

NOTE 11 This address is wrapped into the keyset_block, see 9.3.2.7.2.2.

drm_time – This parameter defines the time in universal time coordinated (UTC). This 40-bit field contains the current time and date in UTC and MJD; see A.3.1.

local_time_offset – This parameter indicates the local time offset from the (UTC) drm_time as defined in A.3.3.

registration_timestamp_start – Indicates from what time onwards the registration data is valid. This is an extra mechanism above the expiration date of the RI certificate. The format of the 40-bit mjdutc field is specified in A.3.1.

NOTE 12 This parameter can also be used against replay attacks.

registration_timestamp_end – Indicates from what time onwards the registration data is expires. This is an extra mechanism above the expiration date of the RI certificate. The format of the 40-bit mjdutc field is specified in A.3.1.

NOTE 13 This parameter can also be used against replay attacks.

shortform_udn – This parameter allows the RI to give an own defined short number identifying the device. This number can be used as a shorter alternative to the UDN during offline notifications. The shortform_udn is coded in BCD format.

signature_block – The signature shall enable a single source authenticity check on the message. The algorithm used for the signature is RSA-1024 or RSA-2048 or RSA-4096. The signature will apply to the implementation guidelines of PKCS#1, as outlined in Clause A.10.

Message result:

The stored RI context shall at a minimum contain

- RI ID, unique device filter (UDF);
- the following keys:
  - UGK, SGK1..n and/or UDK;
  - RIAK;
  - SK.

  If domain addressing via an OMA DRM 2.0 domain is required, the keyset shall (in addition to the standard addressing above) include the following keys:
  - BDK;
  - shortform broadcast domain filter (SBDF), a.k.a. "shortform_domain_id"; see A.11.1.
- For mixed-mode devices the domain context shall additionally contain
  - long-form broadcast domain filter (LBDF), a.k.a. "longform_domain_id()"; see A.11.6.
- A device may have several domain contexts with an RI.
- The RI context shall also contain an RI context expiry time, which is defined to be the timestamp of the registration data if that was send and otherwise the expiration of the RI certificate.
- The RI context may also contain the RI certificate validation data.
- If the RI context has expired, the device shall not execute any other protocol than the 1-pass binary device data registration protocol with the associated RI (context) and upon detection of RI context expiry, the device should initiate the offline notification of detailed device data protocol using the RI_ID stored in the RI context. Depending on the implementation, a dialogue will be shown to the user and the offline NDD protocol will be executed.
  - Accessing a service guide for purchase is still allowed, as this will require a registration first.
  - The device shall be rendered inoperable for any purchase protocol or playback of future content. The device may use stored BCROs to play old content for which the device obtained ROs, but shall not use these BCROs for new content received after the re-registration request until the device is re-registered with the RI.

Requirements:

- The device shall have at most one RI context with each RI. An existing RI context shall be replaced with a newly established RI context after successful re-registration with the same RI.
- The device shall support at least 6 RI contexts for broadcast mode of operation.
- For standard addressing the keyset shall include a valid set of
  - UGK, SGK1..n and/or UDK keys;
  - RIAK key (a single RIAK key is bound to a single group, or, if no UGK nor SGKs have been issued to the device, is bound to a single device);
  - Unique device filter (UDF).

- If domain addressing via an OMA DRM 2.0 domain is required, the keyset shall (in addition to the standard addressing above) include a valid set of
  - BDK key;
  - short-form broadcast domain filter (SBDF), a.k.a. "shortform_domain_id"; see A.11.1.

  And in case of mixed-mode operation devices, the keyset shall contain
  - a long-form broadcast domain filter (LBDF, a.k.a. "longform_domain_id()") that matches the SBDF; see A.11.6.

9.3.2.7.2.2 Protection of the keyset

The structure of the device_registration_response() message is shown in Figure 33. The device_registration_response() message is split in two parts: device-specific (time bound) data and global (not time bound) data.

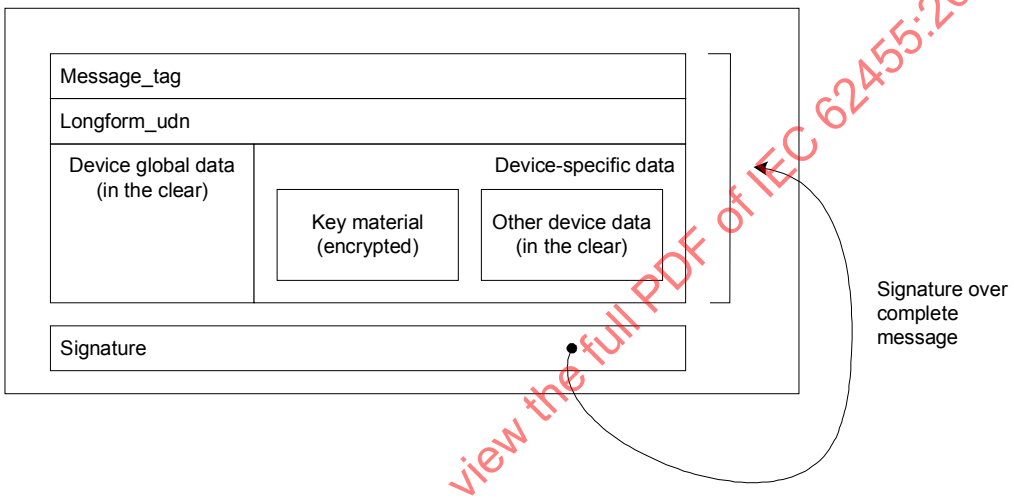


Figure 33 – Device_registration_response() message

The device global data shall be in the clear. The device specific data contains the keyset for the device. This key material shall be encrypted, whereas the rest of the device-specific data shall be in the clear. The key material shall be protected by encryption. The RI shall use the device's public key to encrypt all key material in the device-specific data part of the message.

The RI shall use his private key to sign the complete message data. Upon reception, the device shall verify the RI signature, by using the issuer's public key from the RI certificate. The device shall make sure that this message is correct by using a valid and correct RI certificate.

The complete message shall be authenticated by a signature from the RI.

Creation of the encrypted message shall adhere to the following rules.

a) Generate a (128 or 192 or 256) bit AES key to be used as session key (SK) for the device_registration_response() message.

b) Concatenate the keyset (UGK, SGK1..n, UDK, RIAK, UDF and/or BDK, SBDF plus optional LBDF and CGF if applicable) under rules of FIPS PUB 197:2001 and the tag length format specified in Clause A.11.

c) Determine if the trust authority has defined an algorithm for extra encryption of the keyset_block. If so, prepare the appropriate TAA descriptor for it (see A.11.7). Prepend this to the keyset concatenation of step b).

d) If a TAA_descriptor containing a TAA_algorithm field has been inserted in the keyset_block, encrypt the keyset_block starting at the last bit of the TAA_descriptor while using the algorithm and parameter as indicated in the TAA_descriptor.

e) The concatenated keyset shall be padded with one bit with the value '1' and, after this 1-valued bit, 0 to 63 bits with the value '0', such that the length of the padded keyset is a multiple of 64 bits, see Appendix A of NIST 800-38A. Note that if the non-padded keyset was already a multiple of 64 bits in length, it is padded with 64 bits.

f) Encrypt the keyset using NIST:2001 using the generated SK as (AES-WRAP style) keyset encryption key. This will produce the *keyset_block*.

g) Calculate the part of the keyset_block that would fit into the RSA block (depending on the size of RSA used, be that 1024, 2048 or 4096; see also Clause B.10), including the SK and under implementation rules of the PKCS#1; see A.10.2. If the keyset_block fits into one RSA block, continue at step i). Otherwise, continue at step h).

h) If the SK plus keyset_block including PKCS#1 header, aligning, etc. did not fit into one RSA block, then keep the remainder part as surplus_block().

i) Encrypt SK plus the (part of the) keyset_block that fits into the RSA block with the public key of the target device using RSA (1024 or 2048 or 4096) under implementation guidelines of PKCS#1; see A.10.2. This will produce the *sessionkey_block()*.

j) Concatenate the (non-encrypted) parameters that were not used in the keyset_block and create the message "header" from this. See 9.3.2.7.2.3 for details.

NOTE The sessionkey_block(), the (optional) surplus_block() and the signature_block are not part of the message header).

k) Concatenate the message "header" and the sessionkey_block(). If the SK plus the keyset_block including the PKCS#1 header, aligning, etc. did not fit into one RSA block, then also concatenate surplus_block() part. Hash the result under implementation guidelines of IETF RFC 3447; see Clause A.10. This will produce the signature_input_data.

l) Sign the signature_input_data with RSA (1024 or 2048 or 4096) using the private key of the RI. The signature will apply to the implementation guidelines of PKCS#1, as outlined in Clause A.10. This will produce the *signature_block*.

m) The device_registration_response() message comprises the message "header" plus sessionkey_block(), optionally the surplus_block() and the signature_block; see Figure 34.

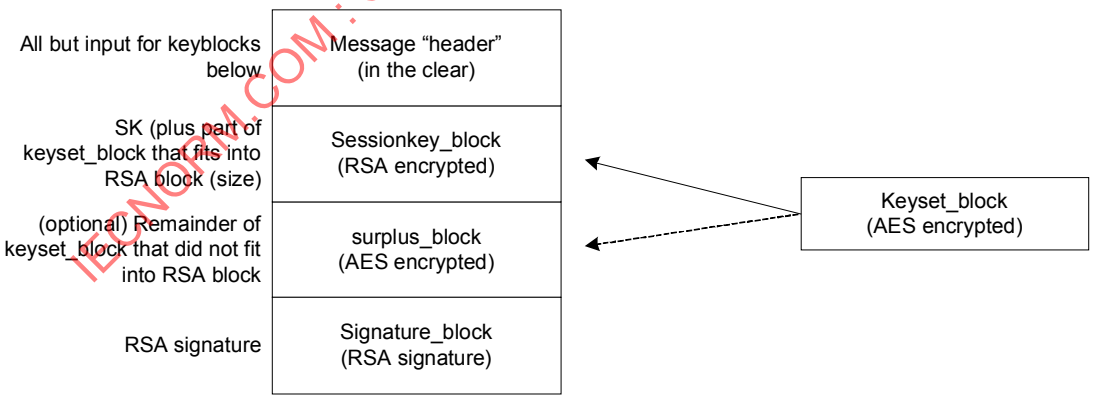


Figure 34 – Structure of device_registration_response() message

In conclusion, the number of RSA blocks used should be kept to a minimum. The AES surplus_block() is present when the keyset does not completely fit into the sessionkey_block() given the RSA block size used. If present the AES surplus_block() contains those keys that did not fit into one RSA block (i.e. the sessionkey_block()). The complete keyset needed for operation after registration is included in the encrypted keyset_block, which is concatenated from the first part in the sessionkey_block() and optionally the surplus_block(). See Clause A.6 for calculations on the surplus_block_size.

Decryption of the encrypted message shall adhere to the following rules.

a) Locate the message via message_tag.

b) Verify if the message is intended for this device by comparing the long_form_udn with the UDN stored in the device.

c) Verify the signature_block of the message by using the public key from the RI.

d) Locate the sessionkey_block() and decrypt the block with the private key of the local device. Locate the session key (SK) from the header and (eventual) padding (according to PKCS#1 and A.10.2). Then locate the keyset_block part from the header and (eventual) padding (according to PKCS#1 and A.10.2). See Clause B.10 for the determination of the session key length.

e) (Optionally) If there is a surplus_block() concatenate this part to the keyset_block. This will complete the keyset_block.

f) Use the SK to decrypt the keyset_block, using NIST:2001.

g) Remove padding from the keyset_block.

h) If, in the keyset block from step g), there is no TAA_descriptor present, go to step i). If there is a TAA_descriptor present, part of the keyset_block was double encrypted. Decrypt the keyset_block (i.e. result from step g)) anew starting at the last bit of the TAA_descriptor while using the algorithm and parameter as indicated in the TAA_descriptor.

i) Allocate the individual keyset_items from the keyset_block according to NIST:2001 and the tag length format specified in Clause A.11.

The SK shall be stored into protected storage. The AES encrypted keyset_block may be stored as is into unprotected storage and decrypted by the device upon use. If the keyset_block is not stored but the decrypted keys from that block are stored instead, the device shall store all key data safely. The keys shall not leak outside the device.

9.3.2.7.2.3 Message syntax

The syntax of device_registration_response() is specified in Table 68.

Table 68 – Message syntax

Field	Length	Type
device_registration_response(){		
/* signature protected part starts here */		
/* message header starts here */		
message_tag	8	bslbf
protocol_version	4	bslbf
reserved_for_future_use	4	bslbf
longform_udn()	80	bslbf
status	8	bslbf
flags {		
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
local_time_offset_flag	1	bslbf
time_stamp_flag	1	bslbf
subscriber_group_key_flag	4	bslbf
shortform_udn_flag	1	bslbf
signature_type_flag	2	bslbf
surplus_block_flag	1	bslbf
keyset_block_length	16	uimsbf

Field	Length	Type
}		
certificate_version	8	bslbf
for(cnt1=0; cnt1 < ri_certificate_counter;cnt1++){		
c_length	16	uimsbf
ri_certificate()	8*c_length	bslbf
}		
for(cnt2=0; cnt2 < ocsp_response_counter;cnt2++){		
r_length	16	uimsbf
ocsp_response()	8*r_length	bslbf
}		
drm_time	40	mjdutc
if (local_time_offset_flag == 0x1) {		
local_time_offset	16	bslbf
}		
if (time_stamp_flag == 0x1) {		
registration_timestamp_start	40	mjdutc
registration_timestamp_end	40	mjdutc
}		
if (shortform_udn_flag == 0x1) {		
shortform_udn	32	bslbf
}		
/* message header ends here */		
if (signature_type_flag == 0x0){		
sessionkey_block()	1024	bslbf
} else if (signature_type_flag == 0x1) {		
sessionkey_block()	2048	bslbf
} else if (signature_type_flag == 0x2) {		
sessionkey_block()	4096	bslbf
}		
if (surplus_block_flag == 0x1){		
surplus_block()	a	bslbf
}		
/* signature protected part ends here */		
if (signature_type_flag == 0x0){		
signature_block	1024	bslbf
} else if (signature_type_flag == 0x1) {		
signature_block	2048	bslbf
} else if (signature_type_flag == 0x2) {		
signature_block	4096	bslbf
}		
}		
[a] For details, see Clauses A.6 and B.10.		

The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this standard.

9.3.2.7.3 (Force to) Re-register – re_register_msg() message

9.3.2.7.3.1 Message specification

Using the 1-pass IRD protocol (see 9.3.2.4) the RI sends a re_register_msg() message, indirectly triggering a (re)registration. The message is specified in Table 69.

Table 69 – Message fields

re_register_msg( )		
Parameter name	(M)andatory/(O)ptional[a]	Remark
message_tag	M	
protocol_version	M	
longform_udn	M	
status	M	
signature_type_flag	M	
certificate_version	M	
ri_certificate_counter	M	
c_length	M	
ri_certificate	M	
ocsp_response_counter	M	
r_length	M	
ocsp_response	M	
signature_block	M	

[a] (O)ptional means that the user of the message may include the parameter in the message, but the device shall support the interpretation of the parameter.
(M)andatory means that the user of the message shall include the parameter in the message.

message_tag – This parameter identifies the type of the message. See Clause A.12 for the value of the message_tag.

protocol_version – This parameter indicates the protocol version of this message. The Device shall ignore messages that have a protocol_version number it does not support. The value of the protocol_version of this message shall be set to 0x0 (i.e. the original format).

If this parameter is set to 0x0, the format specified in this standard is used. If this parameter set to anything else than 0x0, the format is beyond the scope of this version of this standard.

longform_udn() – The long form of the UDN. See 9.3.2.6 for details.

status – The status parameter shall indicate one of the values defined in Table 70. The device shall ignore messages with other error values.

Table 70 – Status values

Status value	Meaning
Success	The message contains valid reregistration message and cancels any preceding forced channel usage restrictions.
ForceInteractiveChannel	If the device is a mixed-mode device, the (re-) registration will be possible via OOB and/or the interactive channel. By using this status code, the RI can indicate to the device that the device shall direct subsequent (re-)registrations to the RI over the device's interactive channel only. When the device receives this status code it will also exclusively use the interactive channel for all other messages. When the interactive channel of the device is not able to connect to the RI the mixed-mode device may revert to the OOB re-registration dialogue. NOTE A mixed-mode device will remain to have full broadcast reception capabilities after receiving this status code.
ForceOobChannel	If the device is a mixed-mode device, the (re-)registration will be possible via OOB and/or the interactive channel. By using this status code, the RI can indicate to the device that the device shall direct subsequent (re-)registrations to the RI over the device's OOB channel. When the device receives this status code, it will also exclusively use the OOB channel for all other messages. NOTE A mixed-mode device will remain to have full interactive channel capabilities after receiving this status code but will not use the interactive channel.

See Clause A.2 for the value of the error codes.

signature_type_flag – A flag to signal type the type of signature algorithm used. See Table 67 for the values allowed and their meaning.

certificate_version – This parameter is a numerical representation of the version of the RI certificate; see Table 71. Using the certificate_version parameter the device can decide if it is needed to update the RI certificate (if it was stored before).

Table 71 – Fields of certificate_version parameter

Parameter field name	Field value ($_h$)	Supports
major_version_number	0x0,..,0xF	MSB4(certificate_version)
minor_version_number	0x0,..,0xF	LSB4(certificate_version)

The parameter is divided into two fields of four bits, whereas the first four bits (msb left) express the major number and the last four bits (lsb right) express the minor version. The major and minor number encode in bslbf format. Sixteen major and 16 minor versions are supported.

EXAMPLE Major.Minor version <1.2> is expressed as 0001 0010_b.

ri_certificate_counter – This parameter indicates the depth of the RI certificate chain; see Table 63.

c_length – This parameter indicates the length in bytes of the ri_certificate.

ri_certificate() – This parameter shall be present. When present, the value of a *ri_certificate* parameter shall be a certificate chain including the RI's certificate. The chain shall not include the root certificate. The RI certificate shall come first in the list. Each following certificate shall directly certify the one preceding it.

The device may store RI certificate verification data indicating that an RI certificate chain has been verified. The purpose of this is to avoid repeated verification of the same certificate chain. The RI certificate verification data stored in this way shall uniquely identify the RI certificate and shall be integrity protected. The device should check if the RI certificate chain received in this parameter corresponds to the stored certificate verification data for this RI. If so, the device need not verify the RI certificate chain again; otherwise the device shall verify the RI certificate chain.

If an RI certificate is received that is not in the stored certificate verification data for this RI, and if the device can determine (in the case of broadcast devices that support DRM time) that the expiry time of the received RI certificate is later than the RI context for this RI, and the certificate status of the RI certificate as indicated in the OCSP response is good (see OCSP-MP), then the device shall verify the complete chain and should replace the stored RI certificate verification data with the received RI certificate data and set the RI context expiry time to that of the received RI certificate expiry time.

However, if the device does store RI certificate verification data in this way it shall store the expiry period of the RI's certificate (as indicated by the notAfter field within the certificate) and shall compare the device's current DRM time with the stored RI certificate expiry time whenever verifying the signature on signed messages from the RI. If the device's current DRM time is after the stored RI certificate expiry time then the device shall abandon processing the RI message and shall initiate the registration protocol.

ocsp_response_counter – This parameter indicates the depth of the OCSP response chain; see Table 64.

r_length – This parameter indicates the length in bytes of the ocsp_response.

ocsp_response() – This parameter, when present, shall be a complete set of valid OCSP responses for the RI's certificate chain. The device shall not fail due to the presence of more than one OCSP response element. A device shall check that an OCSP response is present in the received message. If no OCSP response is present in the device_registration_response() message, then the device shall abort the registration protocol.

signature_block – The signature shall enable a single source authenticity check on the message. The algorithm used for the signature is RSA-1024 or RSA-2048 or RSA-4096. The signature will apply to the implementation guidelines of PKCS#1, as outlined in Clause A.10.

9.3.2.7.3.2 Message syntax

The syntax of re_register_msg() is specified in Table 72.

Table 72 – Message syntax

Field	Length	Type
re_register_msg() {		
/* signature protected part starts here */		
message_tag	8	bslbf
protocol_version	4	bslbf
reserved for future use	4	bslbf
longform_udn()	80	bslbf
status	8	bslbf
flags {		
signature_type_flag	2	bslbf
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
reserved for future use	8	bslbf
}		
certificate_version	8	bslbf
for(cnt1=0; cnt1 < ri_certificate_counter;cnt1++){		
c_length	16	uimsbf
ri_certificate()	8*c_length	bslbf
}		
for(cnt2=0; cnt2 < ocsp_response_counter;cnt2++){		
r_length	16	uimsbf
ocsp_response()	8*r_length	bslbf
}		
/* signature protected part ends here */		
if (signature_type_flag == 0x0){		
signature_block	1024	bslbf
} else if (signature_type_flag == 0x1) {		
signature_block	2048	bslbf
} else if (signature_type_flag == 0x2) {		
signature_block	4096	bslbf
}		
}		

The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this standard.

9.3.2.7.4 Update RI certificate – update_ri_certificate_msg() message

Using the 1-pass IRD protocol (see 9.3.2.4) the RI sends an update_ri_certificate_msg() message, forcing the device to update his RI certificate chain.

This update_ri_certificate_msg() trigger is almost identical to the re_register_msg() message specified in 9.3.2.7.3, with the only adaptation being that the message_tag is different. See Clause A.12 for the value of the message_tag.

9.3.2.7.5 Updating the DRM time – update_drmtime_msg() message

9.3.2.7.5.1 Message specification

Using the 1-pass IRD protocol (see 9.3.2.4), the RI sends an update_drmtime trigger message with the DRM time to the device as specified in Table 73.

Table 73 – Message fields

Update_drmtime_msg( )		
Parameter name	**(M)andatory/(O)ptional[a]**	**Remark**
message_tag	M	
protocol_version	M	
status	M	
signature_type_flag	M	
local_time_offset_flag	M	
drm_time	M	
local_time_offset	O	
signature_block	M	

[a] (O)ptional means that the user of the message may include the parameter in the message, but the device shall support the interpretation of the parameter.
(M)andatory means that the user of the message shall include the parameter in the message.

message_tag – This parameter identifies the type of the message. See Clause A.12 for the value of the message_tag.

protocol_version – This parameter indicates the protocol version of this message. The device shall ignore messages that have a protocol_version number it does not support. The value of the protocol_version of this message shall be set to 0x0 (i.e., the original format).

If this parameter is set to 0x0, the format specified in this version of this standard is used. If this parameter is set to anything else than 0x0, the format is beyond the scope of this standard.

status – The status parameter shall indicate one of the values defined in Table 74. The device shall ignore messages with other error values.

Table 74 – Status values

Status value	Meaning
Success	The message contains valid DRM time RI
NotSupported	The RI does not support the sending of DRM time request. The device will use other means to update DRM time
DeviceTimeError	The RI concluded that the devicetime might be false and forces the device to update its time. As an extra result, the device will determine the clock drift, if any, and notify this to the RI per ARC (off-line notification of short device data; see 9.3.2.3) NOTE This capability should be used with great care.

See Clause A.2 for the value of the error codes.

local_time_offset_flag – Binary flag to signal the presence of the local_time_offset field. See Table 65 for the values allowed and their meaning.

signature_type_flag – A flag to signal the type of signature algorithm used. See Table 67 for the allowed values and their meaning.

drm_time – This parameter defines the DRM time in universal time coordinated (UTC). This 40-bit field contains the current time and date in UTC and MJD; see A.3.1.

local_time_offset – This parameter indicates the local time offset from the (UTC) drm_time as defined in A.3.3.

signature_block – The signature shall enable a single source authenticity check on the message. The algorithm used for the signature is RSA-1024 or RSA-2048 or RSA-4096. The signature will apply to the rules of PKCS#1, as outlined in Clause A.10.

9.3.2.7.5.2 Message syntax

The syntax of update_drmtime_msg() is specified in Table 75.

Table 75 – Message syntax

Field	Length	Type
update_drmtime_msg(){		
/* signature protected part starts here */		
message_tag	8	bslbf
protocol_version	4	bslbf
reserved_for_future_use	4	bslbf
status	8	bslbf
flags {		
local_time_offset_flag	1	bslbf
signature_type_flag	2	bslbf
reserved for future use	5	bslbf
}		
drm_time	40	mjdutc
if (local_time_offset_flag == 0x1) {		
local_time_offset	16	bslbf
}		
/* signature protected part ends here */		
if (signature_type_flag == 0x0){		
signature_block	1024	bslbf
} else if (signature_type_flag == 0x1) {		
signature_block	2048	bslbf
} else if (signature_type_flag == 0x2) {		
signature_block	4096	bslbf
}		
}		

The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this version of this standard.

9.3.2.7.6 Update the contact number – update_contact_number_msg() message

9.3.2.7.6.1 Message specification

Using the 1-pass IRD protocol (see 9.3.2.4) the RI sends an update_contact_number_msg() message with a (set of) contact number(s) to the device as specified in Table 76.

Table 76 – Message fields

Update_contact_number_msg( )		
Parameter name	(M)andatory/(O)ptional[a]	Remark
message_tag	M	
protocol_version	M	
status	M	
signature_type_flag	M	
ri_certificate_counter	M	
c_length	M	
ri_certificate	M	
ocsp_response_counter	M	
r_length	M	
ocsp_response	M	
contact_counter	M	
contact	O	
signature_block	M	

[a] (O)ptional means that the user of the message may include the parameter in the message, but the device shall support the interpretation of the parameter.
(M)andatory means that the user of the message shall include the parameter in the message.

message_tag – This parameter identifies the type of the message. See Clause A.12 for the value of the message_tag.

protocol_version – This parameter indicates the protocol version of this message. The device shall ignore messages that have a protocol_version number it does not support. The value of the protocol_version of this message shall be set to 0x0 (i.e. the original format).

If this parameter is set to 0x0, the format specified in this standard is used. If this parameter is set to anything else than 0x0, the format is beyond the scope of this standard.

status – The status parameter shall indicate one of the values defined in Table 77. The Device shall ignore messages with other error values.

Table 77 – Status values

Status value	Meaning
Success	The message contains valid contact numbers from the RI.
NotSupported	The RI does not support the sending of contact numbers. The device will use other means to use contact numbers (for example, via the service guide).

See Clause A.2 for the value of the error codes.

signature_type_flag – A flag to signal type the type of signature algorithm used. See Table 67 for the allowed values and their meaning.

certificate_version – This parameter is a numerical representation of the version of the RI certificate; see Table 78. Using the certificate_version parameter the device can decide if it is needed to update the RI certificate (if it was stored before).

Table 78 – Fields of certificate_version parameter

Parameter field name	Field value ($_h$)	Supports
major_version_number	0x0,..,0xF	MSB4(certificate_version)
minor_version_number	0x0,..,0xF	LSB4(certificate_version)

The parameter is divided into two fields of four bits, whereas the first four bits (msb left) express the major number and the last four bits (lsb right) express the minor version. The major and minor number encode in bslbf format. Sixteen major and 16 minor versions are supported.

EXAMPLE Major.Minor version <1.2> is expressed as 0001 0010_b.

ri_certificate_counter – This parameter indicates the depth of the RI certificate chain; see Table 63.

c_length – This parameter indicates the length in bytes of the ri_certificate.

ri_certificate() – This parameter shall be present. When present, the value of a *ri_certificate* parameter shall be a certificate chain including the RI's certificate. The chain shall not include the root certificate. The RI certificate shall come first in the list. Each following certificate shall directly certify the one preceding it.

The device may store RI certificate verification data indicating that an RI certificate chain has been verified. The purpose of this is to avoid repeated verification of the same certificate chain. The RI certificate verification data stored in this way shall uniquely identify the RI certificate and shall be integrity protected. The device should check if the RI certificate chain received in this parameter corresponds to the stored certificate verification data for this RI. If so, the device need not verify the RI certificate chain again; otherwise, the device shall verify the RI certificate chain.

If an RI certificate is received that is not in the stored certificate verification data for this RI, and if the device can determine (in the case of broadcast devices that support DRM time) that the expiry time of the received RI certificate is later than the RI context for this RI, and the certificate status of the RI certificate as indicated in the OCSP response is good (see OCSP-MP), then the device shall verify the complete chain and should replace the stored RI certificate verification data with the received RI certificate data and set the RI context expiry time to that of the received RI certificate expiry time.

However, if the device does store RI certificate verification data in this way it shall store the expiry period of the RI's certificate (as indicated by the notAfter field within the certificate) and shall compare the device's current DRM time with the stored RI certificate expiry time whenever verifying the signature on signed messages from the RI. If the device's current DRM Time is after the stored RI certificate expiry time then the device shall abandon processing the RI message and shall initiate the registration protocol.

ocsp_response_counter – This parameter indicates the depth of the OCSP response chain; see Table 64.

r_length – This parameter indicates the length in bytes of the ocsp_response.

ocsp_response() – This parameter, when present, shall be a complete set of valid OCSP responses for the RI's certificate chain. The device shall not fail due to the presence of more than one OCSP response element. A device shall check that an OCSP response is present in

the received message. If no OCSP response is present in the device_registration_response() message, then the device shall abort the registration protocol.

contacts_counter – This parameter indicates the number of contacts carried in the message.

contact – This object specifies the contact, see 9.3.2.7.6.3.

signature_block – The signature shall enable a single source authenticity check on the message. The algorithm used for the signature is RSA-1024 or RSA-2048 or RSA-4096. The signature will apply to the implementation guidelines of PKCS#1, as outlined in Clause A.10.

9.3.2.7.6.2 Message syntax

The syntax of update_contact_number_msg() is specified in Table 79.

Table 79 – Message syntax

Field	Length	Type
update_contact_number_msg() {		
/* signature protected part starts here */		
message_tag	8	bslbf
protocol_version	4	bslbf
reserved_for_future_use	4	bslbf
status	8	bslbf
flags {		
contacts_counter	4	bslbf
reserved for future use	4	bslbf
signature_type_flag	2	bslbf
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
}		
certificate_version	8	bslbf
for(cnt1=0; cnt1 < ri_certificate_counter;cnt1++){		
c_length	16	uimsbf
ri_certificate()	8*c_length	bslbf
}		
for(cnt2=0; cnt2 < ocsp_response_counter;cnt2++){		
r_length	16	uimsbf
ocsp_response()	8*r_length	bslbf
}		
for(cnt3=0; cnt3 < contacts_counter;cnt3++){		
contact()		see 9.3.2.7.6.3
}		
/* signature protected part ends here */		
if (signature_type_flag == 0x0){		
signature_block	1024	bslbf
} else if (signature_type_flag == 0x1) {		
signature_block	2048	bslbf
} else if (signature_type_flag == 0x2) {		

Field	Length	Type
signature_block	4096	bslbf
}		
}		

The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this standard.

9.3.2.7.6.3 Format of the contact object

The format of the contact object is specified in Table 80.

Table 80 – Format of contact object

Field	Length	Type
contact(){		
contact_type	4	uimsbf
reserved for future use	4	bslbf
contact_length	8	uimsbf
contactdata	8*contact_length	bslbf
}		

The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this standard.

contact_type – This field specifies the type of action as listed in Table 81.

Table 81 – Contact_type

Contact_type	Description	Comments	Max length (chars)
0x00	local_ri_phone_number	The number the user of the device needs to contact to start service provision.	20
0x01	int_ri_phone_number	The number the user of the device needs to contact to start service provision when he would call from abroad.	20
0x02	ri_sms_number	The SMS number the user of the device needs to contact to start service provision.	20
0x03	ri_url	The URL address the user of the device needs to contact to start service provision.	30
0x04	local_home_coc_phone_number	The number the user of the device needs to contact to start service provision.	20
0x05	int_home_coc_phone_number	The number the user of the device needs to contact to start service provision when he would call from abroad.	20
0x06	home_coc_sms_number	The SMS number the user of the device needs to contact to start service provision.	20
0x07	home_coc_url	The URL address the user of the device needs to contact start service provision.	30
0x08	local_reporting_phone_number	The number the user of the device needs to contact to report token consumption.	20
0x09	int_reporting_phone_number	The number the user of the device needs to contact to report token consumption when he would call from abroad.	20

Contact_type	Description	Comments	Max length (chars)
0x0A	reporting_sms_number	The SMS number the user of the device needs to contact to report token consumption.	20
0x0B	reporting_url	The URL address the user of the device needs to contact to report token consumption.	30
0x0C-0x0F	Reserved for future use		

contact_length – This parameter indicates the length in bytes of the contact field. The maximum length of the contacts is specified in Table 81.

UTF-8, see IETF RFC 3629, character encoding for ASCII characters is 'efficient' with 1 byte per character. On the other hand, there are characters that are encoded using 6 bytes (Asian languages).

EXAMPLE A URL is limited to 30 characters. The 30 URL UTF-8 characters are translated into bytes as follows.

- "Western" languages – character is 1 byte – Longest URL encoded as bytes is 1*30 characters = 30 bytes.
- Asian languages – character is 6 bytes – Longest URL encoded as bytes is 6*30 characters = 180 bytes.

contactdata – The value in this field specifies any of the contact_type possibilities the user of the Device needs to contact (via other means) to start service provision. For the encoding rules, see Table 82.

Table 82 – Encoding rules for contactdata

Contact types	Contactdata encoding rules
phone numbers	The phone number is encoded as UTF-8, supporting telephone numbers like: "0800-123456789" but also for example: "0800-philips"
SMS numbers	The SMS number is encoded as UTF-8, supporting telephone numbers like: "0800-123456789" but also for example: "philips+subscribe"
URLs	The URL is encoded as UTF-8, according to IETF RFC 1738, supporting URLs like: www.philips.com/start.

9.3.2.7.7 Present TAA report – present_TAA_report() message

Using the 1-pass IRD protocol (see 9.3.2.4) the RI sends a present_TAA_report() message when the RI wants the device to compute the TAA report and show it to the user. See 9.3.2.3.8, A.8.7 and A.11.7 for details on the TAA report.

The present_TAA_report() is identical to the re_register_msg() specified in 9.3.2.7.3, except for the following.

- The value of the message_tag field is as specified in Clause A.12 for the present_TAA_report() message.
- The status field shall always have the value 'Success' from Clause A.2.

9.3.3 Domain joining and leaving

9.3.3.1 General

Interactive devices will adhere to OMA-ERP-DRM-V2_0.

- Interactive devices will therefore use OMA DRM 2.0 domain ID.

Broadcast devices will adhere to the mechanisms as specified in this subclause.

- Broadcast devices will use "shortform_domain_id" a.k.a. SBDF.

Mixed-mode shall have the "interoperability" requirement to support both domain ID formats of interactive and broadcast devices.

- Mixed-mode devices will receive
  - "longform_domain_id()", a.k.a. LBDF, which is a translation of OMA DRM 2.0 domain ID;
  - "shortform_domain_id" a.k.a. SBDF.
- Mixed-mode devices registered for both interactive and broadcast operations may pass either domain ID format to other mixed-mode devices in the domain.
- Interactive only devices shall pass longform_domain_id() format to other devices in the domain. The mixed-mode device will understand this, while broadcast does not understand.
- Broadcast devices shall pass shortform_domain_id format to other devices in the domain. The mixed-mode device will understand this, while interactive does not understand.

9.3.3.2 Protocol overview

The broadcast channel registration layer protocol overview (see 9.3.2.1) results in the specification of several protocols; see Table 83 and Table 84.

Table 83 – Off-line protocols (from device to RI)

Protocol	Subclause	Purpose
Off-line domain join request protocol	9.3.3.3	Request to join a domain
Off-line domain leave request protocol	9.3.3.4	Request to leave a domain

Table 84 – 1-pass protocols (from RI to device)

Protocol	Subclause	Purpose
1-pass binary push device registration protocol	9.3.2.4	Transmit registration data to device
1-pass binary inform registered device protocol	9.3.2.5	Inform device via messages

These protocols interrelate in the way (roundtrip), as specified in Table 85.

Table 85 – Protocol interrelation

Kicking off action...	...results in
Off-line domain join request (request to join a domain)	domain_registration_response() message (transmit registration data to device)
Off-line domain leave request (request to leave a domain)	domain_update_response() message (inform device via messages)
join_domain_msg() (inform device via messages)	Off-line domain join request, which in turn may result in domain_registration_response() as listed above
leave_domain_msg() (inform device via messages)	Off-line domain leave request, which in turn may result in domain_update_response() as listed above

9.3.3.3 Off-line domain join request

When the user of a device might want to join a particular domain, he uses the NSD protocol with the destined action code range (see 9.3.2.3).

9.3.3.4 Off-line domain leave request

When the user of a device might want to leave a particular domain, he uses the NSD protocol with the destined action code range (see 9.3.2.3).

9.3.3.5 Binary messages

9.3.3.5.1 Domain data – domain_registration_response() message

9.3.3.5.1.1 Message specification

Using the 1-pass PDR protocol (see 9.3.2.4) the RI sends a domain_registration_response() message, informing the device of a new domain keyset. The message is specified in Table 86.

Table 86 – Message fields

Domain_registration_response()		
Parameter name	(M)andatory/(O)ptional[a]	Remark
message_tag	M	Global, not encrypted
protocol_version	M	Global, not encrypted
longform_udn	M	Global, not encrypted
device_nonce	M	Device specific, not encrypted
status	M	Device specific, not encrypted
time_stamp_flag	M	Device specific, not encrypted
certificate_version	M	Global, not encrypted
ri_certificate_counter	M	Global, not encrypted
c_length	M	Global, not encrypted
ri_certificate	M	Global, not encrypted
ocsp_response_counter	M	Global, not encrypted
r_length	M	Global, not encrypted
ocsp_response	M	Global, not encrypted
domain_timestamp_start	O	Device specific, not encrypted
domain_timestamp_end	O	Device specific, not encrypted
signature_type_flag	M	Global, not encrypted
keyset_block_length	M	Device specific, not encrypted
TAA_descriptor	O	Device specific, encrypted
broadcast_domain_key	M	Device specific, encrypted
longform_domain_id()	O	Device specific, encrypted
shortform_domain_id	M	Device specific, encrypted
signature_block	M	Device specific, not encrypted

[a] (O)ptional means that the user of the message may include the parameter in the message, but the device shall support the interpretation of the parameter.
(M)andatory means that the user of the message shall include the parameter in the message.

message_tag – This parameter identifies the type of the message. See Clause A.12 for the value of the message_tag.

protocol_version – This parameter indicates the protocol version of this message. The device shall ignore messages that have a protocol_version number it does not support. The value of the protocol_version of this message shall be set to 0x0 (i.e. the original format).

If this parameter is set to 0x0, the format specified in this standard is used. If this parameter is set to anything else than 0x0, the format is beyond the scope of this standard.

longform_udn() – The long form of the UDN. See 9.3.2.6 for details.

status – The status parameter shall indicate one of the values defined in Table 87. The device shall ignore messages with other error values.

Table 87 – Status values

Status value	Meaning
Success	The message contains valid domain registration data from the RI.
NotSupported	The RI does not support the sending of domain registration data from the RI. The RI shall not include any valid keyset in the message. The device will use other means to obtain valid domain registration data from the RI.
InvalidDomain	The RI could not recognize the domain identifier that was used in the join domain request or decided that the domain identifier is invalid. The RI shall not include any valid keyset in the message.
DomainFull	The RI indicates that no more devices are allowed to join the domain. The RI shall not include any valid keyset in the message.

See Clause A.2 for the value of the error codes.

device_nonce – The device_nonce is the nonce that was present in the request (using the offline NSD protocol) to which this message is a response. This nonce is an encoded in BCD.

time_stamp_flag – Binary flag to signal the presence of the domain_timestamp_start and domain_timestamp_end field. See Table 65 for the allowed values and their meaning.

certificate_version – This parameter is a numerical representation of the version of the RI certificate; see Table 88. Using the certificate_version parameter the device can decide if it is needed to update the RI certificate (if it was stored before).

Table 88 – Fields of certificate_version parameter

Parameter field name	Field value ($_h$)	Supports
major_version_number	0x0,..,0xF	MSB4(certificate_version)
minor_version_number	0x0,..,0xF	LSB4(certificate_version)

The parameter is divided into two fields of four bits, whereas the first four bits (msb left) express the major number and the last four bits (lsb right) express the minor version. The major and minor number encode in bslbf format. Sixteen major and 16 minor versions are supported.

EXAMPLE Major.Minor version <1.2> is expressed as 0001 0010_b.

ri_certificate_counter – This parameter indicates the depth of the RI certificate chain, see Table 63.

c_length – This parameter indicates the length in bytes of the ri_certificate.

ri_certificate() – This parameter shall be present. When present, the value of a *ri_certificate* parameter shall be a certificate chain including the RI's certificate. The chain shall not include

the root certificate. The RI certificate shall come first in the list. Each following certificate shall directly certify the one preceding it.

The device may store RI certificate verification data indicating that an RI certificate chain has been verified. The purpose of this is to avoid repeated verification of the same certificate chain. The RI certificate verification data stored in this way shall uniquely identify the RI certificate and shall be integrity protected. The device should check if the RI certificate chain received in this parameter corresponds to the stored certificate verification data for this RI. If so, the device need not verify the RI certificate chain again; otherwise the device shall verify the RI certificate chain.

If an RI certificate is received that is not in the stored certificate verification data for this RI, and if the device can determine (in the case of broadcast devices that support DRM time) that the expiry time of the received RI certificate is later than the RI context for this RI, and the certificate status of the RI certificate as indicated in the OCSP response is good (see OCSP-MP), then the device shall verify the complete chain and should replace the stored RI certificate verification data with the received RI certificate data and set the RI context expiry time to that of the received RI certificate expiry time.

However, if the device does store RI certificate verification data in this way it shall store the expiry period of the RI's certificate (as indicated by the notAfter field within the certificate) and shall compare the device's current DRM time with the stored RI certificate expiry time whenever verifying the signature on signed messages from the RI. If the device's current DRM time is after the stored RI certificate expiry time then the device shall abandon processing the RI message and shall initiate the registration protocol.

ocsp_response_counter – This parameter indicates the depth of the OCSP response chain; see Table 64.

r_length – This parameter indicates the length in bytes of the ocsp_response.

ocsp_response() – This parameter, when present, shall be a complete set of valid OCSP responses for the RI's certificate chain. The device shall not fail due to the presence of more than one OCSP response element. A device shall check that an OCSP response is present in the received message. If no OCSP response is present in the domain_registration_response() message, then the device shall abort the registration protocol.

domain_timestamp_start – Indicates from what time onwards the registration data for the domain is valid. This is an extra mechanism above the expiration date of the RI certificate. The format of the 40-bit mjdutc field is specified in A.3.1.

NOTE 1 This parameter can also be used against replay attacks.

domain_timestamp_end – Indicates from what time onwards the registration data for the domain expires. This is an extra mechanism above the expiration date of the RI certificate. The format of the 40-bit mjdutc field is specified in A.3.1.

NOTE 2 This parameter can also be used against replay attacks.

signature_type_flag – A flag to signal type the type of signature algorithm used. See Table 67 for the values allowed and their meaning.

keyset_block_length – This parameter indicates the length in bits of the total keyset_block. That is the part in the sessionkey_block().

TAA_descriptor – This is the TAA descriptor as specified in A.11.7.

NOTE 3 This key is wrapped into the keyset_block; see 9.3.3.5.1.2.

broadcast_domain_key – An AES symmetric key to address a broadcast domain. This key is also known as BDK. The key length shall be 128 bits.

NOTE 4 This key is wrapped into the keyset_block; see 9.3.2.7.2.2.

longform_domain_id() – This parameter is also known as the long-form broadcast domain filter (LBDF). See A.11.6 for its definition. The longform_domain_id() is used for mixed-mode operation.

NOTE 5 This address is wrapped into the keyset_block; see 9.3.2.7.2.2.

shortform_domain_id – This parameter is also known as the short-form broadcast domain filter (SBDF). See A.11.1. An addressing scheme used to filter for messages like BCROs. The shortform_domain_id is used for broadcast mode of operation.

NOTE 6 This address is wrapped into the keyset_block; see 9.3.2.7.2.2.

signature_block – The signature shall enable a single source authenticity check on the message. The algorithm used for the signature is RSA-1024 or RSA-2048 or RSA-4096. The signature will apply to the implementation guidelines of PKCS#1, as outlined in Clause A.10.

Message result:

The stored domain context shall at a minimum contain

- Following keys:
  - BDK;
  - short-form broadcast domain filter (SBDF), a.k.a. "shortform_domain_id"; see A.11.1.
- For mixed-mode operation, devices' domain context shall additionally contain
  - long-form broadcast domain filter (LBDF), a.k.a. "longform_domain_id()"; see A.11.6.
- A device may have several domain contexts with an RI.
- If the domain context has expired, the device shall not execute any other protocol than the 1-pass binary device data registration protocol with the associated RI (context), and upon detection of domain context expiry, the device should initiate the offline notification of short device data protocol using the correct ARC. Depending on the implementation, a dialogue will be shown to the user and the offline NSD protocol will be executed.
  - Accessing a service guide for purchase is still allowed, as this will require a (domain) registration first.
  - The device shall be rendered inoperable for any purchase protocol or playback of future content. The device may use stored BCROs to play old content for which the device obtained ROs, but shall not use these BCROs for new content received after the re-registration request until the device is re-registered with the RI.

Requirements:

- If domain addressing via an OMA DRM 2.0 domain is required the keyset shall (additionally to the standard addressing as defined in 9.3.2.7.2.2) include a valid set of
  - BDK key;
  - short-form broadcast domain filter (SBDF), a.k.a. "shortform_domain_id"; see A.11.1.

  And in the case of mixed-mode operation devices, the keyset shall contain
  - a long-form broadcast domain filter (LBDF, a.k.a. "longform_domain_id()") that matches the SBDF; see A.11.6.

9.3.3.5.1.2 Protection of the keyset

The structure of the domain_registration_response() message is shown in Figure 35. The domain_registration_response() message is split in two parts: device-specific (time bound) data and global (not time bound) data.

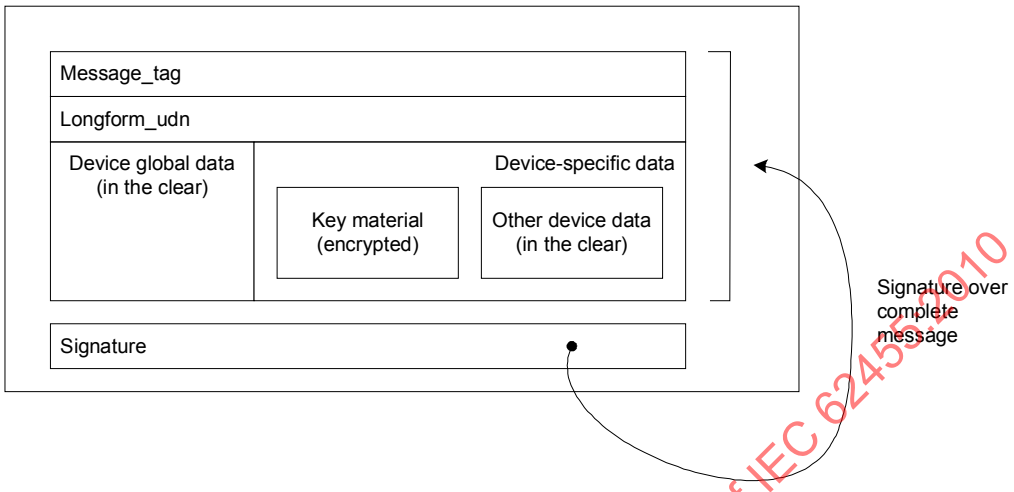


Figure 35 – Domain_registration_response() message

The device global data shall be in the clear. The device-specific data contains the keyset for the device. This key material shall be encrypted, whereas the rest of the device-specific data shall be in the clear. The key material shall be protected by encryption. The RI shall use the device's public key to encrypt all key material in the device-specific data part of the message.

The RI shall use his private key to sign the complete message data. Upon reception, the device shall verify the RI signature, by using the issuer's public key from the RI certificate. The device shall make sure that this message is correct by using a valid and correct RI certificate.

The complete message shall be authenticated by a signature from the RI.

Creation of the encrypted message shall adhere to the following rules.

a) Generate a (128 or 192 or 256) bit AES key to be used as session key (SK) for the domain_registration_response() message.

b) Determine if the trust authority has defined an algorithm for extra encryption of the keyset_block. If so, prepare the appropriate TAA descriptor item for it (see A.11.7).

c) Concatenate the TAA descriptor item from step b) if it is present, the keyset (BDK, SBDF plus optional LBDF if applicable) according to the rules of FIPS PUB 197:2001 and the tag length format specified in Clause A.11. The keyset may contain multiple domain contexts (i.e. a matching BDK, SBDF and optionally LBDF) but not more than the maximum indicated size of the single RSA block used for the sessionkey_block.

d) If a TAA descriptor containing a TAA_algorithm field has been inserted in the keyset_block, encrypt the keyset_block starting at the last bit of the TAA_descriptor while using the algorithm and parameter as indicated in the TAA_descriptor.

NOTE 1 The TAA descriptor length is a multiple of 64 bits plus 1 bit, see A.11.7.

e) The concatenated keyset shall be padded with one bit with the value '1' and, after this 1-valued bit, 0 to 63 bits with the value '0', such that the length of the padded keyset is a multiple of 64 bits, see Appendix A of NIST 800-38A. Note that if the non-padded keyset was already a multiple of 64 bits in length, it is padded with 64 bits.

f) Encrypt the keyset using NIST:2001 using the generated SK as (AES-WRAP style) keyset encryption key. This will produce the *keyset_block*.

g) Calculate the part of the keyset_block that would fit into the RSA block (depending on the size of RSA used, be that 1024, 2048 or 4096; see also Clause B.10), including the SK and under implementation rules of the PKCS#1; see A.10.2.

h) Encrypt the SK plus the keyset_block with the public key of the target device, using RSA (1024 or 2048 or 4096) under the implementation guidelines of PKCS#1; see A.10.2. This will produce the *sessionkey_block()*.

i) Concatenate the (non-encrypted) parameters that were not used in the keyset_block and create the message "header" from this. See 9.3.3.5.1 for details.

NOTE 2 The sessionkey_block() and the signature_block are not part of the message header.

j) Concatenate the message "header" and the sessionkey_block(). Hash the result under implementation guidelines of IETF RFC 3447; see Clause A.10. This will produce the signature_input_data.

k) Sign the signature_input_data with RSA (1024 or 2048 or 4096) using the private key of the RI. The signature will apply to the implementation guidelines of PKCS#1, as outlined in Clause A.10. This will produce the *signature_block*.

l) The domain_registration_response() message comprises the message "header" plus sessionkey_block() and the signature_block; see Figure 36.

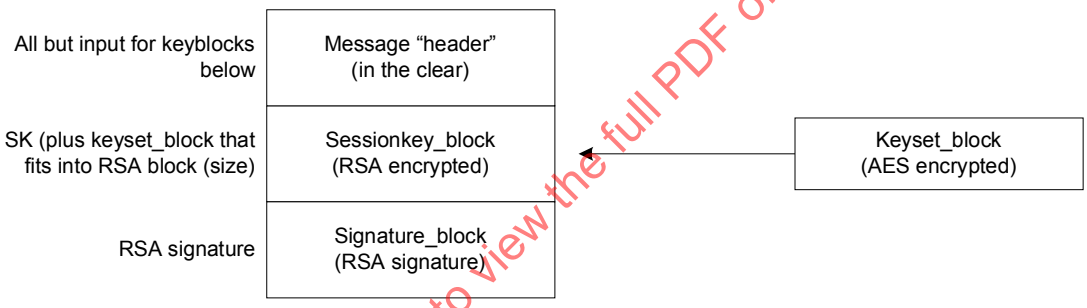


Figure 36 – Structure of domain_registration_response() message

Decryption of the encrypted message shall adhere to the following rules.

a) Locate the message via message_tag.

b) Verify if the message is intended for this device by comparing the long_form_udn with the UDN stored in the device.

c) Verify the signature_block of the message by using the public key from the RI.

d) Locate the sessionkey_block() and decrypt the block with the private key of the local device. Locate the session key (SK) from the header and (eventual) padding (according to PKCS#1 and A.10.2). Then locate the keyset_block part from the header and (eventual) padding (according to PKCS#1 and A.10.2). See Clause B.10 for the determination of the session key length.

e) Use the SK to decrypt the keyset_block, using NIST:2001.

f) Remove padding from the keyset_block.

g) If, in the keyset block from step f), there is no TAA_descriptor present, go to step h). If there is a TAA_descriptor present, part of the keyset_block was double encrypted. Decrypt the keyset_block (i.e. result from step f)) anew starting at the last bit of the TAA_descriptor while using the algorithm and parameter as indicated in the TAA_descriptor.

h) Allocate the individual keyset_items from the keyset_block according to NIST:2001 and the tag length format specified in Clause A.11.

The SK shall be stored into protected storage. The AES encrypted keyset_block may be stored as is into unprotected storage and decrypted by the device upon use. If the

keyset_block is not stored but the decrypted keys from that block are stored instead, the device shall store all key data safely. The keys shall not leak outside the device.

9.3.3.5.1.3 Message syntax

The syntax of the domain_registration_response() message is specified in Table 89.

Table 89 – Message syntax

Field	Length	Type
domain_registration_response(){		
/* signature protected part starts here */		
/* message header starts here /*		
message_tag	8	bslbf
protocol_version	4	bslbf
reserved_for_future_use	4	bslbf
unique_device_number	80	bslbf
reserved_for_future_use	4	bslbf
device_nonce	4	bslbf
status	8	bslbf
flags {		
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
signature_type_flag	2	bslbf
time_stamp_flag	1	bslbf
reserved for future use	7	bslbf
keyset_block_length	16	uimsbf
}		
certificate_version	8	bslbf
for(cnt1=0; cnt1 < ri_certificate_counter;cnt1++){		
c_length	16	uimsbf
ri_certificate()	8*c_length	bslbf
}		
for(cnt2=0; cnt2 < ocsp_response_counter;cnt2++){		
r_length	16	uimsbf
ocsp_response()	8*r_length	bslbf
}		
if (time_stamp_flag == 0x1) {		
domain_timestamp_start	40	mjdutc
domain_timestamp_end	40	mjdutc
}		
/* message header ends here /*		
if (signature_type_flag == 0x0){		
sessionkey_block()	1024	bslbf
} else if (signature_type_flag == 0x1) {		
sessionkey_block()	2048	bslbf
} else if (signature_type_flag == 0x2) {		

Field	Length	Type
sessionkey_block()	4096	bslbf
}		
/* signature protected part ends here */		
if (signature_type_flag == 0x0){		
signature_block	1024	bslbf
} else if (signature_type_flag == 0x1) {		
signature_block	2048	bslbf
} else if (signature_type_flag == 0x2) {		
signature_block	4096	bslbf
}		
}		

The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this standard.

9.3.3.5.2 Updating a domain – domain_update_response() message

9.3.3.5.2.1 Message specification

Using the 1-pass IRD protocol (see 9.3.2.5), the RI sends a domain_update_response() message, informing the device that it left a particular domain. The message is specified in Table 90.

Table 90 – Message fields

Domain_update_response()		
Parameter name	(M)andatory/(O)ptional[a]	Remark
message_tag	M	Global, not encrypted
protocol_version	M	Global, not encrypted
longform_udn	M	Global, not encrypted
status	M	Device specific, not encrypted
device_nonce	M	Device specific, not encrypted
certificate_version	M	Global, not encrypted
ri_certificate_counter	M	Global, not encrypted
c_length	M	Global, not encrypted
ri_certificate	M	Global, not encrypted
ocsp_response_counter	M	Global, not encrypted
r_length	M	Global, not encrypted
ocsp_response	M	Global, not encrypted
shortform_domain_id	M	Device specific, not encrypted
signature_type_flag	M	Global, not encrypted
signature_block	M	Device specific, not encrypted

[a] (O)ptional means that the user of the message may include the parameter in the message, but the device shall support the interpretation of the parameter.
(M)andatory means that the user of the message shall include the parameter in the message

message_tag – This parameter identifies the type of the message. See Clause A.12 for the value of the message_tag.

protocol_version – This parameter indicates the protocol version of this message. The device shall ignore messages that have a protocol_version number it does not support. The value of the protocol_version of this message shall be set to 0x0 (i.e. the original format).

If this parameter is set to 0x0, the format specified in this standard is used. If this parameter is set to anything else than 0x0, the format is beyond the scope of this standard.

longform_udn() – The long form of the UDN. See 9.3.2.6 for details.

status – The status parameter shall indicate one of the values defined in Table 91. The device shall ignore messages with other error values.

Table 91 – Status values

Status value	Meaning
Success	The message informs the device that the RI has removed this device from the domain in which it was registered. The device shall remove the domain keyset that was associated to the particular domain.
NotSupported	The RI does not support the request to leave a domain. The device will use other means to notify the RI that it wants to leave a particular domain.
InvalidDomain	The RI is unable to support the request to leave a domain, because the domain is invalid.

See Clause A.2 for the value of the error codes.

device_nonce – The device_nonce is the nonce that was present in the request (using the offline NSD protocol) to which this message is a response. This nonce is an encoded in BCD.

certificate_version – This parameter is a numerical representation of the version of the RI certificate; see Table 92. Using the certificate_version parameter, the customer device can decide if it is needed to update the RI certificate (if it was stored before).

Table 92 – Fields of certificate_version parameter

Parameter field name	Field value ($_h$)	Supports
major_version_number	0x0,..,0xF	MSB4(certificate_version)
minor_version_number	0x0,..,0xF	LSB4(certificate_version)

The parameter is divided 2 fields of 4 bits, whereas the first 4 bits (msb left) express the major number and the last four bits (lsb right) express the minor version. The major and minor number encoded in bslbf format. Sixteen major and 16 minor versions are supported.

EXAMPLE Major.Minor version <1.2> is expressed as 0001 0010_b.

ri_certificate_counter – This parameter indicates the depth of the RI certificate chain; see Table 63.

c_length – This parameter indicates the length in bytes of the ri_certificate.

ri_certificate() – This parameter shall be present. When present, the value of a *ri_certificate* parameter shall be a certificate chain including the RI's certificate. The chain shall not include the root certificate. The RI certificate shall come first in the list. Each following certificate shall directly certify the one preceding it.

The device may store RI certificate verification data indicating that an RI certificate chain has been verified. The purpose of this is to avoid repeated verification of the same certificate chain. The RI certificate verification data stored in this way shall uniquely identify the RI certificate and shall be integrity protected. The device should check if the RI certificate chain received in this parameter corresponds to the stored certificate verification data for this RI. If

so, the device need not verify the RI certificate chain again; otherwise, the device shall verify the RI certificate chain.

If an RI certificate is received that is not in the stored certificate verification data for this RI, and if the device can determine (in the case of broadcast devices that support DRM time) that the expiry time of the received RI certificate is later than the RI context for this RI, and the certificate status of the RI certificate as indicated in the OCSP response is good (see OCSP-MP), then the device shall verify the complete chain and should replace the stored RI certificate verification data with the received RI certificate data and set the RI context expiry time to that of the received RI certificate expiry time.

However, if the device does store RI certificate verification data in this way, it shall store the expiry period of the RI's certificate (as indicated by the notAfter field within the certificate) and shall compare the device's current DRM time with the stored RI certificate expiry time whenever verifying the signature on signed messages from the RI. If the device's current DRM time is after the stored RI certificate expiry time then the device shall abandon processing the RI message and shall initiate the registration protocol.

ocsp_response_counter – This parameter indicates the depth of the OCSP response chain; see Table 64.

r_length – This parameter indicates the length in bytes of the ocsp_response.

ocsp_response() – This parameter, when present, shall be a complete set of valid OCSP responses for the RI's certificate chain. The device shall not fail due to the presence of more than one OCSP response element. A device shall check that an OCSP response is present in the received message. If no OCSP response is present in the domain_registration_response() message, then the device shall abort the registration protocol.

shortform_domain_id – The shortform_domain_id is the SBDF.

signature_type_flag – A flag to signal type the type of signature algorithm used. See Table 67 for the values allowed and their meaning.

signature_block – The signature shall enable a single source authenticity check on the message. The algorithm used for the signature is RSA-1024 or RSA-2048 or RSA-4096. The signature will apply to the implementation guidelines of PKCS#1, as outlined in Clause A.10.

9.3.3.5.2.2 Message syntax

The syntax of domain_update_response() is specified in Table 93.

Table 93 – Message syntax

Field	Length	Type
domain_update_response(){		
/* signature protected part starts here */		
message_tag	8	bslbf
protocol_version	4	bslbf
reserved_for_future_use	4	bslbf
longform_udn()	80	bslbf
reserved_for_future_use	4	bslbf
device_nonce	4	bslbf
status	8	bslbf
flags {		
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
signature_type_flag	2	bslbf
}		
certificate_version	8	bslbf
for(cnt1=0; cnt1 < ri_certificate_counter;cnt1++){		
c_length	16	uimsbf
ri_certificate()	8*c_length	bslbf
}		
for(cnt2=0; cnt2 < ocsp_response_counter;cnt2++){		
r_length	16	uimsbf
ocsp_response()	8*r_length	bslbf
}		
shortform_domain_id	48	uimsbf
/* signature protected part ends here */		
if (signature_type_flag == 0x0){		
signature_block	1024	bslbf
} else if (signature_type_flag == 0x1) {		
signature_block	2048	bslbf
} else if (signature_type_flag == 0x2) {		
signature_block	4096	bslbf
}		
}		

The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this standard.

9.3.3.5.3 (Force to) Join a domain – join_domain_msg() message

Using the 1-pass IRD protocol (see 9.3.2.5), the RI sends a join_domain_msg() message, forcing the device to join a particular domain.

This join_domain_msg() trigger is almost identical to the re_register_msg() message specified in 9.3.2.7.3, with the only adaptation being that the message_tag is different. See Clause A.12 for the value of the message_tag.

9.3.3.5.4 (Force to) Leave a domain – leave_domain_msg() message

9.3.3.5.4.1 Message specification

Using the 1-pass IRD protocol (see 9.3.2.5), the RI sends a leave_domain_msg() message, forcing the device to leave a particular domain.

This leave_domain_msg() trigger is almost identical to the re_register_msg() message specified in 9.3.2.7.3, with the only adaptations being that

- the message_tag is different. See Clause A.12 for the value of the message_tag;
- the shortform_domain_id is incorporated, which is the SBDF.

For the message specification with an explanation of the parameters see the re_register_msg() message. For sake of completion the complete leave_domain_msg() message syntax is defined below.

9.3.3.5.4.2 Message syntax

The syntax of leave_domain_msg() is specified in Table 94.

Table 94 – Message syntax

Field	Length	Type
leave_domain_msg() {		
/* signature protected part starts here */		
message_tag	8	bslbf
protocol_version	4	bslbf
reserved_for_future_use	4	bslbf
longform_udn()	80	bslbf
flags {		
signature_type_flag	2	bslbf
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
reserved for future use	8	bslbf
}		
shortform_domain_id	48	uimsbf
certificate_version	8	bslbf
for(cnt1=0; cnt1 < ri_certificate_counter;cnt1++){		
c_length	16	uimsbf
ri_certificate()	8*c_length	bslbf
}		
for(cnt2=0; cnt2 < ocsp_response_counter;cnt2++){		
r_length	16	uimsbf
ocsp_response()	8*r_length	bslbf
}		
/* signature protected part ends here */		
if (signature_type_flag == 0x0){		
signature_block	1024	bslbf
} else if (signature_type_flag == 0x1) {		
signature_block	2048	bslbf
} else if (signature_type_flag == 0x2) {		
signature_block	4096	bslbf
}		
}		

The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this version of this standard.

9.3.4 Token handling

9.3.4.1 Protocol overview

The theory of operation (see Clause A.16) results in the specification of several protocols; see Table 95 and Table 96.

Table 95 – Offline protocols (from device to RI)

Protocol	Subclause	Purpose
Token request protocol	9.3.4.2	Request to purchase tokens
Token reporting protocol	9.3.4.3	Protocol to report the consumption of tokens

Table 96 – 1-pass protocols (from RI to device)

Protocol	Subclause	Purpose
1-pass binary push device registration protocol	9.3.2.4	Transmit registration data to device
1-pass binary inform registered device protocol	9.3.2.5	Inform device via messages

These protocols interrelate in the way (roundtrip), as specified in Table 97.

Table 97 – Protocol interrelation

Kicking off action...	...results in
Token request protocol (request to purchase tokens)	Token delivery response message (transmit tokens to device)
Token reporting protocol (report the consumption of tokens)	Token delivery response message (transmit tokens to device)

9.3.4.2 Token request protocol

When the user of a device wants to obtain tokens, he uses the NSD protocol with the token_request action type (see 9.3.2.3).

9.3.4.3 Token reporting protocol

When the user of a device is instructed by his device to report token consumption, he uses the NSD protocol with the token_consumption_message action type in order to send a token consumption report (see 9.3.2.3.5).

9.3.4.4 Delivering tokens – token_delivery_response() message

9.3.4.4.1 Message specification

Using the 1-pass PDR protocol (see 9.3.2.5), the RI sends a token_delivery_response() message, informing the device of the delivery of new tokens. The message is specified in Table 98.

Table 98 – Fields of token delivery response message

Token_delivery_response()		
Parameter name	(M)andatory/(O)ptional[a]	Remark
message_tag	M	Not encrypted
protocol_version	M	Not encrypted
message_length	M	Not encrypted
group_size_flag	M	Not encrypted
address_mode	M	Not encrypted
one	M	Not encrypted
address	M	Not encrypted
bit_access_mask	Not used in this standard	Not encrypted
position_in_group	M	Not encrypted
domain_id_extension	Not used in this standard	Not encrypted
domain_generation	Not used in this standard	Not encrypted
rights_issuer_id	M	Not encrypted
status	M	Not encrypted
device_nonce	M	Not encrypted
response_flag	M	Not encrypted
token_reporting_flag	M	Not encrypted
earliest_reporting_time_flag	M	Not encrypted
latest_reporting_time_flag	M	Not encrypted
token_quantity_flag	M	Not encrypted
token_delivery_response_id	M	Not encrypted
latest_consumption_time	O	Not encrypted
earliest_reporting_time	O	Not encrypted
latest_reporting_time_flag	O	Not encrypted
encrypted_token_quantity	O	Encrypted
encrypted_report_authentication_key	O	Encrypted
MAC	M	Not encrypted

[a] (O)ptional means that the user of the message may include the parameter in the message, but the device shall support the interpretation of the parameter.
(M)andatory means that the user of the message shall include the parameter in the message.

message_tag – This parameter identifies the type of the message. See Clause A.12 for the value of the message_tag.

protocol_version – This parameter indicates the protocol version of this message. The device shall ignore messages that have a protocol_version number it does not support. The value of the protocol_version of this message shall be set to 0x0 (i.e. the original format).

If this parameter is set to 0x0, the format specified in this version of this standard is used. If this parameter is set to anything else than 0x0, the format is beyond the scope of this standard.

message_length – 12-bit field indicating the length in bytes of the message starting immediately after this field.

group_size_flag – 1-bit field indicating the group size used. 0 – a maximum group size 256 is used, 1 – a maximum group size of 512 is used.

address_mode – Three-bit field indicating the addressing mode used by this message. Table 99 lists all possible address modes. Not that not all modes are used in this standard for the token delivery response message.

Table 99 – Address_mode for token delivery response message

address_mode	Description
0x0	Addressing whole of unique group This addressing mode is not used in this standard for the token delivery response message.
0x1	Addressing of subscriber group using a bit_mask size of 256 or 512 bit depending on group_size_flag (subset of unique group) This addressing mode is not used in this standard for the token delivery response message.
0x2-0x3	Addressing of unique device
0x4	Addressing of OMA DRM 2.0 domain. Address field concatenated with the domain_id_extension will be the domain_id in this case This addressing mode is not used in this standard for the token delivery response message.
0x5-0x7	Reserved for future use.

one – 1-bit flag that shall have the value 0x1 in this standard. This field may have value 0x0 in future editions of this standard.

address – Four-byte group address. Each rights issuer has its own address space. If the group_size is 512, the group address is made of the first 31 bits of the address field. If this message is addressed to a unique device in a group, the least significant bit of the address field is the most significant bit of the group position.

bit_access_mask – This field is not used in this standard and may be used in future editions. It is indicated here, so devices according to this edition of this standard know its size. All bits of the field shall be set to 0, when the field is not used.

position_in_group – If this message addresses a unique device, this field specifies the position of the unique device in the given subscriber group. If group_size_flag is 0, the position in the group is directly given by the position_in_group field. If group_size_flag is 1, 9 bits are used to identify the position in the group. If group_size_flag is 1, the least significant bit (bit 0) from the address field is used as the 9th bit, the most significant bit. The real position in the group is then given by:

```
int real_position_in_group;
if(address_mode&0x6==0x2){
  if(group_size_flag == 0){
    //maximum size of 256 devices in group.
    real_position_in_group = position_in_group;
  }else{
    //maximum size of 512 devices in group;
    real_position_in_group = ((address&0x1)<<8)|position_in_group;
  }
```

}

domain_id_extension – This field is not used in this standard and may be used in future editions. It is indicated here, so devices according to this edition of this standard know its size. All bits of the field shall be set to 0, when the field is not used.

domain_generation – This field is not used in this standard and may be used in future editions. It is indicated here, so devices according to this edition of this standard know its size. All bits of the field shall be set to 0, when the field is not used.

rights_issuer_id() – The ID of the rights issuer. This is the 160-bit SHA-1 hash of the public key of the RI. See X509SPKIHash in OMA-TS-DRM-DRM-V2_0.

status – The status parameter shall indicate one of the values defined in Table 100. The device shall ignore messages with other error values.

Table 100 – Message error codes

Status value	Meaning
Success	The message contains valid token delivery data from the RI
NotSupported	The RI does not support the sending of tokens from the RI. In this message, the RI shall set the value of token_quantity to zero or shall set the token_quantity_flag to 0x0.
TokenConsumptionMessageError	The RI did receive a token consumption message, but that it was erroneous and that the device should redo the last token consumption message In this token delivery response message, the RI shall set the value of token_quantity to zero or shall set the token_quantity_flag to 0x0. The RI shall use a token_reporting_flag of value 0x1. The RI shall use the device_nonce of the last token consumption message that the RI successfully processed or set the response_flag to 0x0 in case no token consumption messages have been successfully processed. The device shall generate a token consumption message, reporting on the token consumption from the time of the generation of the token consumption message with the same device_nonce as the device_nonce in this token delivery response message, or from first start-up in case the response_flag was set to 0x0.
NoTokenConsumptionMessage	The RI did not receive a token consumption message yet, but was expecting one, because the present date/time is later than the last latest_token_consumption_time sent to the device in a token delivery response message In this token delivery response message, the RI shall set the value of token_quantity to zero or shall set the token_quantity_flag to 0x0. The RI shall use a token_reporting_flag of value 0x1. The RI shall use the device_nonce of the last token consumption message that the RI successfully processed or set the response_flag to 0x0 in case no token consumption messages have been successfully processed. The device shall generate a token consumption message, reporting on the token consumption from the time of the generation of the token consumption message with the same device_nonce as the device_nonce in this token delivery response message, or from first start-up in case the response_flag was set to 0x0.

See Clause A.2 for the value of the error codes.

device_nonce – If the response_flag equals 0x1, the device_nonce is the nonce present in the request (using the offline NSD protocol) to which this token delivery response message is a response. If the response_flag field equals 0x0, this token delivery response message does not refer to any request from the Device to the RI and the device_nonce may be ignored. The nonce is encoded in BCD.

response_flag – If this flag equals 0x1, this token delivery response message is a response to a message from the device to the RI and the device_nonce in this token delivery response message is taken from that message. If this flag equals 0x0, this token delivery response

message does not refer to any message from the device to the RI and the device_nonce can be any value.

token_reporting_flag – If this flag equals 0x1, the device has to report to the RI the consumption of the tokens received with this token delivery response message. If this flag equals 0x0, the device can consume all tokens delivered with this token delivery response message, as well as any other previously delivered tokens that are still not consumed, without ever having to report their consumption.

earliest_reporting_time_flag – Binary flag to signal the presence of the earliest_reporting_time_flag field. See Table 65 for the allowed values and their meaning.

latest_reporting_time_flag – Binary flag to signal the presence of the latest_reporting_time_flag field. See Table 65 for the values allowed and their meaning.

token_quantity_flag – Binary flag to signal the presence of the token_quantity_flag field. See Table 65 for the allowed values and their meaning.

token_delivery_response_id – This is the ID of the token delivery response message. The RI shall use the same token_delivery_response_id when retransmitting a token delivery response message. The RI shall generate a random number using a sufficiently good pseudo random number generator for every new token delivery response message. Devices shall discard token delivery response messages with a token_delivery_response_id identical to the one in an already received token delivery response message.

latest_token_consumption_time – After the date/time indicated in the latest_token_consumption_time field, the device shall not use any tokens, which have been received after the last token delivery response message that had the token_reporting_flag set to 0x0, for the consumption of protected content controlled by the RI. The device shall use the date/time in the latest_token_consumption_time field, if present, of the last received token delivery response message, regardless of the value of the field status. The format of the 40-bit mjdutc field is specified in A.3.1.

earliest_reporting_time – If the device reports the consumption of tokens before the date/time indicated in the earliest_reporting_time field, the RI need not change the latest_token_consumption_time in its subsequent token delivery response message. The format of the 40-bit mjdutc field is specified in A.3.1.

latest_reporting_time – The purpose of this field is to make uninterrupted token consumption possible. If the device reports the token consumption before the date/time indicated in the latest_reporting_time field, the RI shall send the next token delivery response message before the latest_token_consumption_time, unless the RI wishes to interrupt or disable the token consumption. The format of the 40-bit mjdutc field is specified in A.3.1.

encrypted_token_quantity – A 4-byte field, containing the encrypted token_quantity. Token_quantity is a signed, two's complement 32-bit number. If the value of token_quantity is positive, it specifies the number of tokens the device receives from the RI. If the value of token_quantity is negative, it specifies how many tokens the RI removes from the device. If the field encrypted_token_quantity is not present, no tokens are received from the RI and no tokens are removed from the device by this token delivery response message. The token_quantity is encrypted using AES-128-CBC, with fixed IV 0 and with 0 padding in the last block if needed. The encryption key used depends on the addressing mode of the token delivery response message; see Table 101.

Table 101 – Mapping of address_mode to keys for the token delivery response message

Address_mode	Key(s) used to decrypt field
0x0 (unique group)	This addressing mode is not used in this standard for the token delivery response message.
0x1 (subscriber group)	This addressing mode is not used in this standard for the token delivery response message.
0x2-0x3 (unique device)	Token delivery key.
0x4 (domain)	This addressing mode is not used in this standard for the token delivery response message.
0x5-0x7	Reserved for future use.

encrypted_report_authentication_key – This field contains the encrypted report authentication key. The report authentication key a 128-bit key to authenticate the reported number of tokens with in the next token consumption message. The encrypted_report_authentication_key field is only present if the token_reporting_flag has the value 0x1. The RI shall generate a random number using a sufficiently good pseudo random number generator for the value of every newly required report authentication key. The report authentication key is encrypted using AES-128-CBC, with fixed IV 0 and with 0 padding in the last block if needed. The encryption key used depends on the addressing mode of the token delivery response message; see Table 102.

Table 102 – Mapping of address_mode to keys for the token delivery response message

Address_mode	Key(s) used to decrypt field
0x0 (unique group)	This addressing mode is not used in this standard for the token delivery response message.
0x1 (subscriber group)	This addressing mode is not used in this standard for the token delivery response message.
0x2-0x3 (unique Device)	Token delivery key.
0x4 (domain)	This addressing mode is not used in this standard for the token delivery response message.
0x5-0x7	Reserved for future use.

MAC – This is the authentication code calculated over all bytes before this field in this message using HMAC-SHA-1-96 (see IETF RFC 2104). The MAC is used for the integrity check of this message. The key used to create the MAC is the token delivery response message authentication key TDRMAK as defined in A.8.6. Devices shall not use token delivery response messages with an invalid MAC.

Message result:

- More information on device actions after the reception of this message can be found in A.16.2.

9.3.4.4.2 Message syntax

The syntax of the token delivery response message is specified in Table 103.

Table 103 – Syntax of token delivery response message

Field	Length	Type
token_delivery_response(){		
/* MAC protected part starts here */		
message_tag	8	bslbf
protocol_version	4	bslbf
message_length	12	uimsbf
group_size_flag	1	bslbf
reserved for future use	3	bslbf
address_mode	3	uimsbf
one	1	bslbf
address	32	uimsbf
if(address_mode == 0x1 && group_size_flag == 0){ [a]		
bit_access_mask	256	bslbf
}else if(address_mode == 0x1 && group_size_flag == 1){		
bit_access_mask [a]	512	bslbf
}else if (address_mode&0x6 == 0x2){		
position_in_group	8	uimsbf
}else if (address_mode == 0x4){ [a]		
domain_id_extension	6	bslbf
domain_generation	10	uimsbf
}		
rights_issuer_id()	160	bslbf
status	8	bslbf
device_nonce	4	bslbf
flags {		
response_flag	1	bslbf
token_reporting_flag	1	bslbf
earliest_reporting_time_flag	1	bslbf
latest_reporting_time_flag	1	bslbf
token_quantity_flag	1	bslbf
reserved for future use	7	bslbf
}		
token_delivery_response_id	96	bslbf
if (token_reporting_flag == 0x1) {		
latest_token_consumption_time	40	mjdutc
if (earliest_reporting_time_flag == 0x1) {		
earliest_reporting_time	40	mjdutc
}		
if (latest_reporting_time_flag == 0x1) {		
latest_reporting_time	40	mjdutc
}		
}		
/* encrypted part starts here		
if(token_quantity_flag == 1){		

Field	Length	Type
encrypted_token_quantity	32	bslbf
}		
encrypted_report_authentication_key	128	bslbf
/* encrypted part ends here		
/* MAC protected part ends here */		
MAC	96	bslbf
}		
The fields reserved_for_future_use are reserved for future use and all their bits shall be set to 0 in systems according to this standard.		
a Although this addressing mode is indicated here to facilitate future upgrades, this addressing mode is NOT used in this standard for the token delivery response message.		

9.3.5 Mixed-mode registration for interactive and broadcast modes of operation

This subclause applies to devices supporting both communication via a broadcast channel and an interactivity cannel. If such devices are registered for both interactive and broadcast mode of operation, the RI has the option of sending ROs either over a broadcast channel or an interactivity channel, whichever the RI finds better at the time of sending the ROs.

Since the registration of a broadcast-only device involves more user interaction than the registration for an interactive device (see 9.3.2), the registration of a mixed-mode device (with both support for broadcast channel and interactivity channel) shall start with the registration for interactive mode of operation; see Figure 37. Steps 3-10 in this figure are the 4-pass ROAP as specified in OMA-ERP-DRM-V2_0 with which the registration for the interactive mode of operation is done when OMA DRM 2.0 is used on registration layer. In Figure 37, the registration for the interactive mode of operation (steps 3 – 10) was triggered by the reception of a ROAP trigger (step 2), which was sent by the SoC when it received a purchase request (step 1) from a device that was not registered yet. The RI may decide to use the ROAP protocol as specified in OMA XBS 7.3, for sending the registration data for the broadcast mode of operation to mixed-mode devices in step 10.

After successful registration for interactive mode of operation, the device may also join a domain (steps 11 and 12), if so desired.

After step 12, a device that is capable of broadcast operation and uses OMA DRM 2.0 for registration, but that did not receive registration data for the broadcast mode of operation in step 10, shall put itself into registration mode, in which it waits for the registration data for broadcast mode of operation in the form of the device_registration_response() message (step 14 in Figure 37). If the device does not receive the registration data within a timeout the device leaves registration mode and stops listening for device_registration_response() messages.

As part of step 1, the RI obtains the capabilities of the device that wanted to register in the form of the signed XML data of the purchase request, see 12.2.4.5. From these capabilities data, the RI finds out that the device is also capable of broadcast mode of operation. In such a case, the RI may decide to send this device the device_registration_response() message (step 14 in Figure 37), which message contains the registration data for the broadcast mode of operation.

In case the device indicates capability of mixed-mode operation and when the RI wants to include the domain registration in the device registration data as well, this registration data shall include the longform_domain_id().

Part of the registration data for the broadcast mode of operation is the longform_udn(); see 9.3.2.6, which is stored in the device. In the mixed-mode registration outlined above, the RI

will obtain in step 1 the longform_udn() of the device as part of the XML purchase data. See 12.2.4.5 for details on the use of this XML structure.

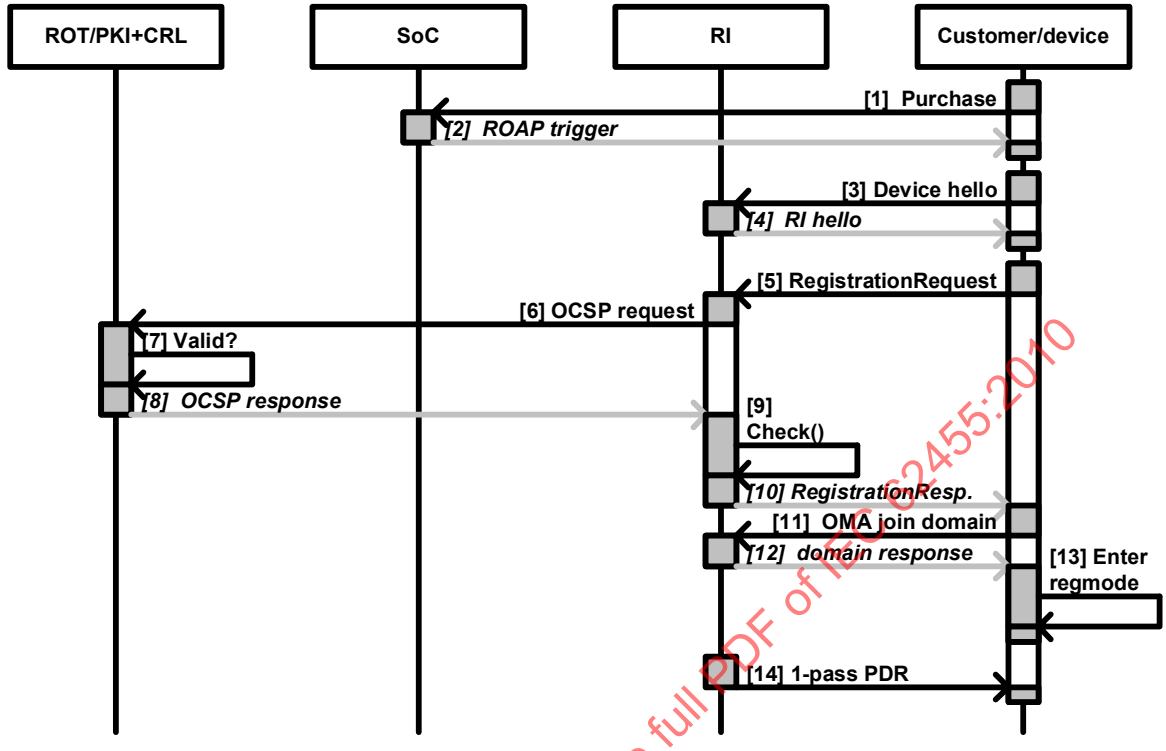


Figure 37 – Registration for mixed-mode operation with one ROT

10 Signalling and service guide

10.1 General

This clause specifies the general requirements and data model for signalling and service and content discovery for any system implementing this standard. Specific information for the following systems can be found as referenced below.

- IPDC over DVB-H, see 13.6;
- DVB-T/C/S, see 14.6;
- MPEG2-TS-based IP systems (15.7):
  - according to DVB-IPI, see ETSI EN 102 034 and other TV-anytime based systems (see 15.7.2);
  - others, using, for example, proprietary signalling and discovery, see 15.7.3;
- non-MPEG2-TS based IP systems (16.6):
  - TV-anytime based systems;
  - others, using, for example, as of yet non-standardized signalling and discovery.

The reason for describing separately requirements and implementations is that, especially for IP systems, standards for signalling and service guides are not well established. While the use of standardized systems is recommended, it is recognized that this standard may also be of use where other signalling and service guides are already in place.

10.2 Signalling requirements

10.2.1 Signalling information

This standard requires some signalling information to be carried alongside protected content, so that devices are able to locate the relevant messages and streams in the received data.

This subclause sets out the requirements and data model for signalling in any system implementing this standard. These requirements apply to all systems and serves as an introduction to the following subclauses. It is expected that existing signalling schemes in most systems will are extensible enough to support these requirements.

10.2.2 Requirements for signalling the KSM

It shall be possible to

- associate a KSM stream with a service, such that one KSM applies to all components in that service;
- for systems that allow the separate encryption of components of a service (for example, audio, video or subtitles), associate a KSM stream with a particular component in a service;
- and signal an access description for the KSM, so that a device is able to locate the stream for a particular service.

Devices shall be able to buffer the access description, in order to ensure quick service access without need for service guide acquisition.

Therefore, the access description can only contain parameters that are likely to change very infrequently for a particular service, so that it can be tolerated that in case of a change, the device performs service guide acquisition before accessing a service.

Multiple KSM streams may be associated with a single service. If those KSM streams cannot be distinguished by IPDCoperatorID (mapping to socID or cocID), another method of associating the KSM stream with the correct rights object (ICRO or BCRO) is to use the most significant 8 bits of the service_CID_extension / programme_CID_extension for this purpose. Those values shall be unique in the different KSM streams for this method to work. In order to find the correct KSM stream, the device will look for a matching srvCIDExt or prgCIDExt attribute (see 13.6.2) in the SDP file.

10.2.3 Requirements for signalling of services

It shall be possible to signal for each service the socID and serviceBaseCID, that is the static part of all CIDs of SEAKs and PEAKs pertaining to the service.

10.3 Service guide requirements

The requirements and data model for service and content discovery via a service guide are specified in 12.5.

10.4 Service guide recommendations

The following RECOMMENDATION is made regarding the use of all service guides.

When describing an item available for purchase, the seller should include some indication of the rights that are being offered. This means that the implications of the constraints and permissions that will be contained in the rights object(s) delivered should be clear.

There follows an incomplete list of examples.

- If a rights object will contain only an ACCESS_ACTION (see Clause A.23), it should be indicated to the user that they will not be able to perform "PVR"-like functions on the content.
- If a rights object will contain an EXPORT_ACTION (see OMA-TS-DRM-REL-V2_0), it should be indicated to the user that they will be able to export the content. Restrictions on the export should also be described.
- If a rights object will contain a PLAY_ACTION with a count constraint, the number of plays allowed should be clearly indicated to the user.
- If a rights object will contain a PLAY_ACTION with a time constraint, the limits of the time constraint should be clearly indicated to the user.

If a service guide includes methods for indicating this type of information, they should be filled in as appropriate. However, current service guides do not generally include fields to indicate this kind of information. In this case, it is RECOMMENDED that a text description is included instead.

11 Rights issuer services and rights issuer streams

11.1 General

In this clause, the rights issuer services and rights issuer streams are specified. The specific use of the key stream layer for the individual supported systems is further specified in 13.7, 14.9, 15.8 and 16.7.

A rights issuer service is a logical service that delivers registration and rights management layer messages to devices over a broadcast channel. The messages carried include all the messages that are allocated a message tag in Clause A.12. A rights issuer service consists of one or more rights issuer streams, which carry the messages and objects. For IPDC over DVB-H, it is also possible to add a rights issuer stream to other types of service, alongside audio and video streams.

RI services and streams are used in

- IPDC over DVB-H systems;
- DVB-T/C/S systems.

In both these systems it may also be possible to deliver rights objects and registration data via an interactivity channel where such a channel is available and when supported by the device.

11.2 Rights issuer services

11.2.1 Requirements for rights issuer services in IPDC over DVB-H systems

Table 104 shows the requirements for the support of rights issuer services and streams by devices in IPDC over DVB-H systems.

Table 104 – Requirements for the support of RI services and streams by IPDC over DVB-H devices

Device type	Support
Interactive device	Not applicable
Mixed-mode device	Mandatory
Broadcast device	Mandatory

Table 105 shows the requirements for the support of rights issuer services and streams by service providers in IPDC over DVB-H systems.

Table 105 – Requirements for the support of rights issuer services and streams by service providers in IPDC over DVB-H systems

Scenario	Support
Network supports only interactive devices	Not applicable
Network supports only interactive and mixed-mode devices	Optional
Network supports broadcast devices (and optionally other types)	Mandatory

11.2.2 Requirements for rights issuer services in DVB-T/C/S systems

All DVB-T/C/S networks using this standard shall support rights issuer services and streams.

All DVB-T/C/S devices using this standard shall support rights issuer services and streams.

11.2.3 Requirements for the support of rights issuer services and streams in IPTV systems

RI services and streams are not used in

- MPEG2 TS-based IP systems;
- non-MPEG2 TS-based IP systems.

Clause 11 does not apply to these systems.

11.3 Usage of rights issuer streams and services

11.3.1 General

This standard aims to allow enough flexibility for operators to fulfil their own requirements for message and rights object delivery, and to trade off latency against bandwidth, while also allowing devices to minimize power consumption. To support this, there are no restrictions on which messages can be carried in which type of stream, although the expected mode of operation is described in Clause B.4, and the following uses of an RI stream are described:

- *ad hoc* RI stream;
- scheduled RI stream;
- in-band RI stream (for IPDC over DVB-H systems only).

However, rights issuers may use rights issuer streams to deliver messages in any way they require that is compliant with this standard.

An informative schedule may be broadcast for RI services. Where available, this shall be provided as part of the service guide. It is used to indicate times at which data for particular sets of devices or subscriber groups will be broadcast. This allows devices to listen to RI services only when necessary and will also allow service operation centres to make use of spare network capacity when available, for example, at night.

Where a rights issuer broadcasts a complete schedule covering all its registered devices, it may have any number of rights issuer services. This schedule shall indicate, for each device or group of devices, a single rights issuer service, which will be used to deliver objects to that set of devices. Otherwise, rights issuers shall have exactly one rights issuer service. This requirement allows a device to determine exactly one rights issuer service to which it listens.

Figure 38 shows how RI streams are carried in RI services and A/V services.

Figure 38 – Relationship between RI service and RI streams and other services and RI streams

Within this subclause, the objects and messages to be carried are referred to as "objects".

11.3.2 Scheduled RI stream

In a scheduled RI stream, the timing of message broadcast may be scheduled in some way, according to device or subscriber groups.

The schedule describes, for each RI service, blocks of times at which messages are expected to be available for particular ranges of devices or subscriber groups. Where provided, it shall be available in the service guide (see Clause A.17).

Although the schedule applies to the whole RI service, it may be that there are streams within the service that do not follow the schedule – for example, *ad hoc* RI streams.

It is recommended that a rights issuer fulfil the advertised schedule. However, when circumstances require, a rights issuer may deviate from the schedule that has been broadcast. This may cause some devices to miss schedule slots.

A scheduled RI service does not have to be available continuously. It could, for example, only be broadcast at night. It is also possible for an RI service's bandwidth to vary.

11.3.3 *Ad hoc* RI stream

An *ad hoc* stream is used to carry messages that a rights issuer wishes to be sent spontaneously, i.e. with low latency. It is expected that a device will receive this stream when it is in some special registration mode or when the rights issuer service is specifically selected.

11.3.4 In-band RI streams within a media service

In band RI streams apply only to IPDC over DVB-H systems. This subclause applies only to such systems. In this subclause, "in-band" means that the stream is logically part of a media service, and will be transmitted in the same time sliced bursts as that service. This allows devices to receive the stream without increasing the length of time for which they must receive the stream, which would cause them to increase their power consumption.

Each protected service may contain in band RI streams. When receiving a protected service, which has associated in band RI streams, a device shall listen to the RI streams for rights issuers with which it is registered when receiving the protected service.

It is expected that in-band RI streams will be used to deliver:

- messages that need to be delivered with low latency to large numbers of devices. Examples include rights objects for free previews of the media service or free-to-view services; or
- rights objects for content being carried by the service.

Devices shall be able to identify in-band RI streams within protected services from the service guide.

12 Service subscription and purchase

12.1 General

Figure 39 and Figure 40 show message flows for service subscription and purchase for the connected and unconnected mode of operation.

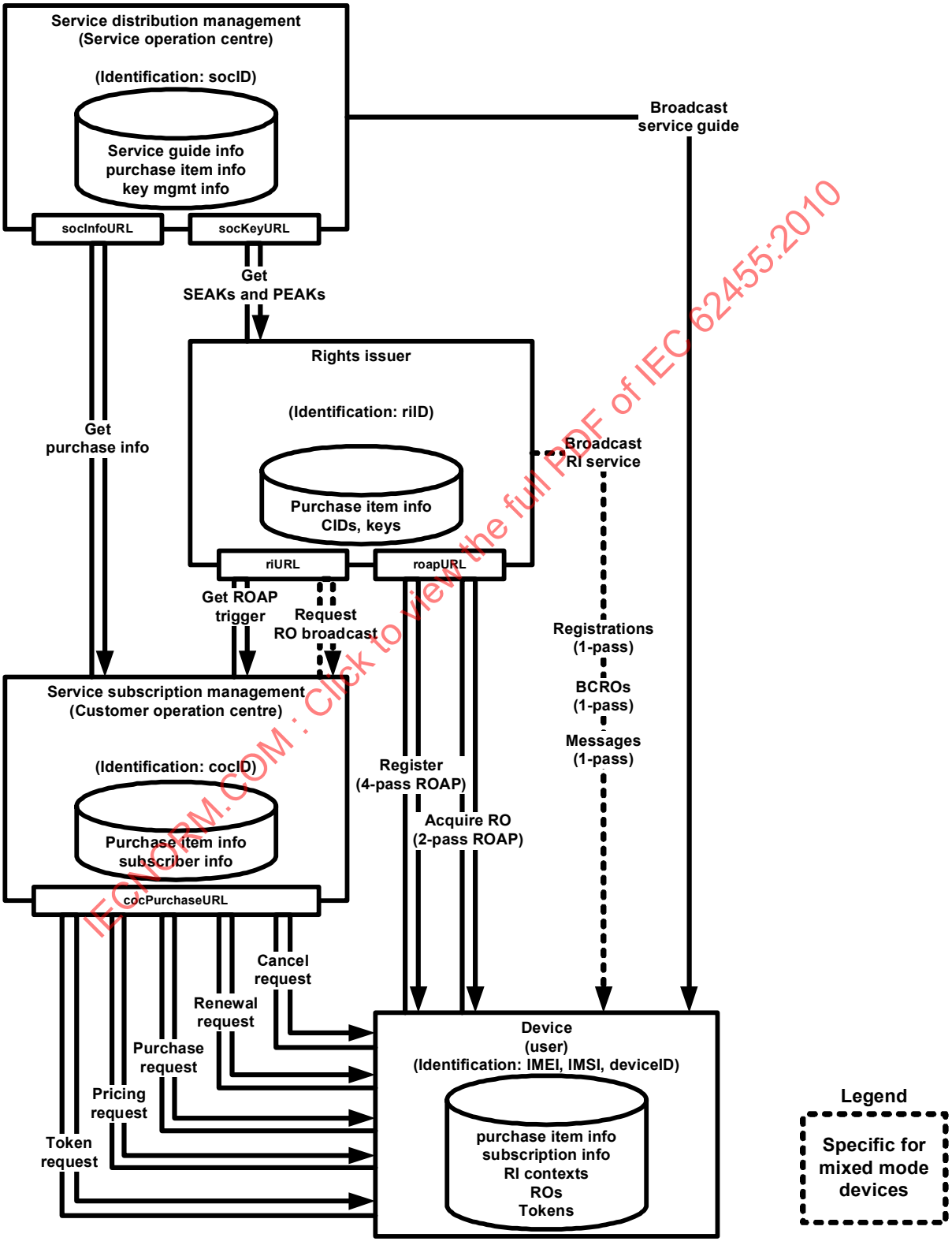


Figure 39 – Message flows for service subscription and purchase for the connected mode of operation

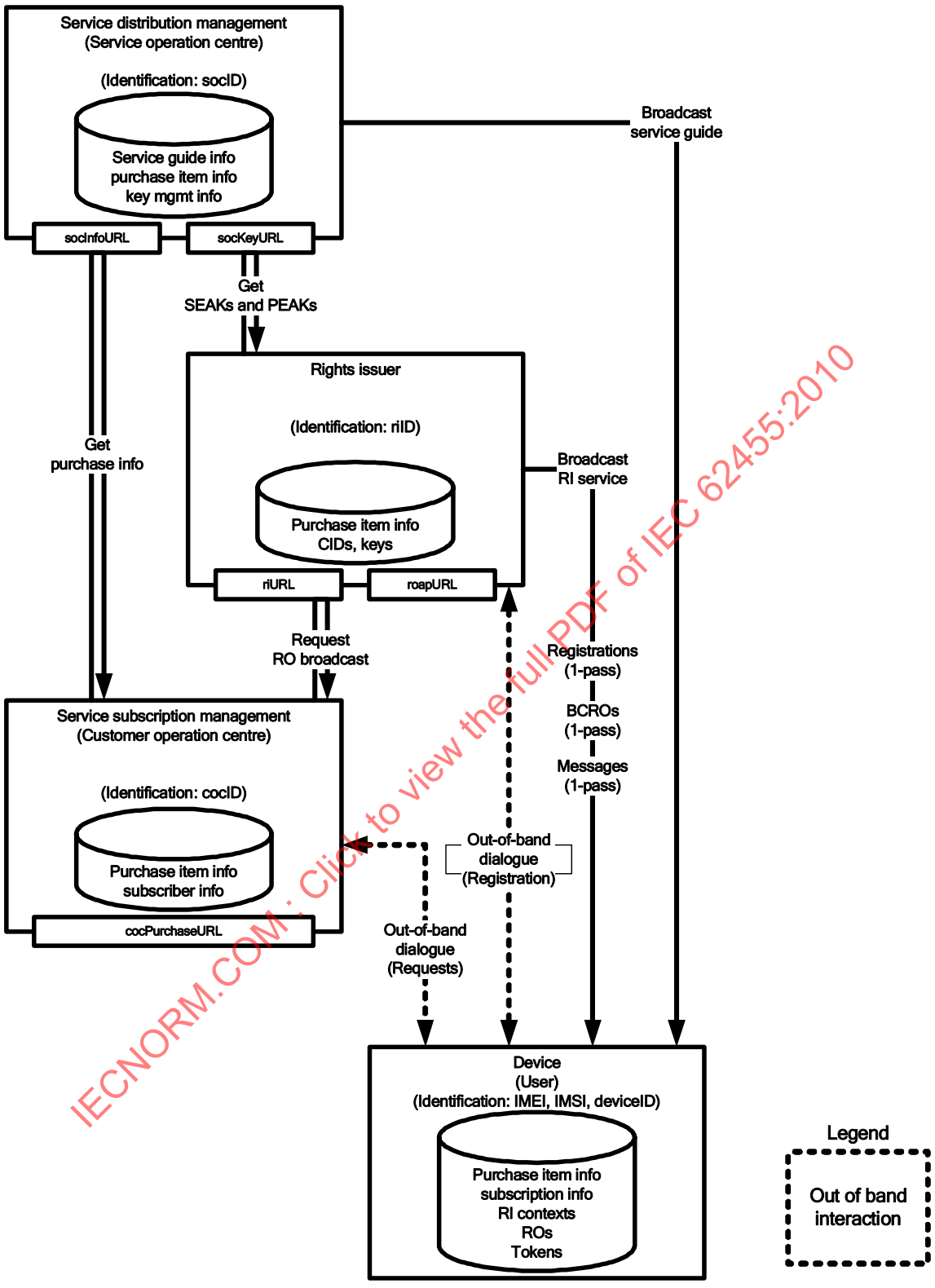


Figure 40 – Message flows for service subscription and purchase for the unconnected mode of operation

12.2 Purchase over an interactivity channel

12.2.1 General

Interactive devices may use an interactivity channel for purchase transactions. The protocol and messages of these purchase transactions are specified later in this subclause.

A device supporting purchase over an interactivity channel shall implement the protocol for communication with the network elements (SOC, COC, RI) defined in this subclause. However, the messages exchanged between network elements are listed here only as an informative reference.

Messages specified in this subclause apply also to mixed-mode devices. However, for mixed-mode devices the RI may choose to deliver a BCRO instead of an ICRO (see 12.3).

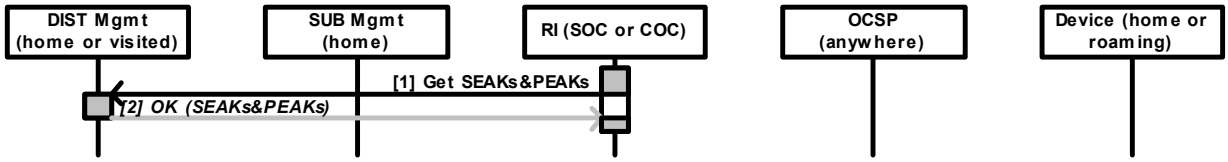
12.2.2 Typical purchase sequences

12.2.2.1 General

Entity definitions for interaction diagrams	
DIST Mgmt	**Service distribution management, part of service operation centre (SOC)** The distribution management system is the system from which the device is receiving broadcast services, and is therefore always local to the device's current position, whether the device is located at home or roaming.
SUB Mgmt	**Service subscription management, part of customer operation centre (COC)** The user is assumed to have a contractual relationship with a home COC (there are no restrictions regarding how many COCs the user may have a contractual relationship with; in case there is more than one, the user needs to choose from which COC to purchase access to a particular service). The operators of the home COC and the SOC (home or visited) need to have a contract ("roaming agreement") in place.
RI	**Rights issuer (can be part of SOC or COC)** The rights issuer is the server-side DRM implementation and can be assumed to be part of either the SOC or part of the COC (from the architectural point of view, no assumption shall be made, and also a stand-alone rights issuer implementation should be enabled).
OCSP	**OCSP responder** The OCSP responder is the external certification authority that is able to certify DRM-related device credentials.
Device	**Device** The device is assumed to be a mobile broadcast device with interaction capabilities (interactive device). The device is further assumed to have an OMA DRM 2.0 compliant DRM agent, which supports the broadcast extensions as specified in this standard. If it is a device supporting mixed-mode registration, it is also assumed to be able to receive BCROs via a broadcast channel.

12.2.2.2 Bulk download of service and programme keys

In cases where the RI is part of the SOC, the download of service and programme keys from the key generator within service distribution management to the RI may be implemented as a periodical "bulk download"; see Figure 41.

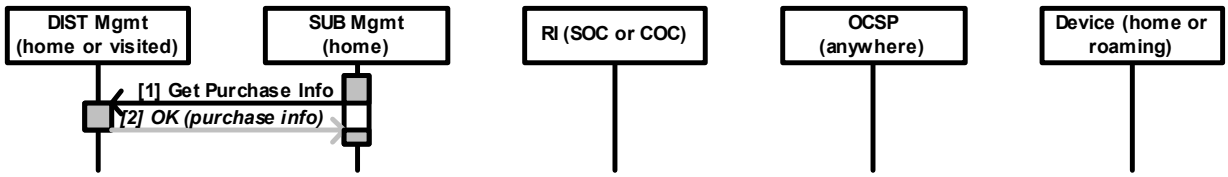


1	**Get SEAKs and PEAKs** The SEAKs and PEAKs are downloaded from the service distribution management to the RI, together with the purchase item identification. SEAKs and PEAKs have a time span of validity (indicating during which time interval they are effectively used to protect broadcast traffic). Each SEAK or PEAK can be associated with one or more purchase items and each purchase item can be associated with multiple SEAKs or PEAKs. With each purchase item, there may be some information regarding usage rules specified, which influence the generation of the rights expression by the RI. Recommended content of the message: • time interval for which associations of purchase items and their respective SEAKs and PEAKs should be returned. *This message is network-internal and is not further specified.*
2	**OK (SEAKs and PEAKs)** The answer to a 'Get SEAKs and PEAKs' request contains a set of associations between purchase item identification and the corresponding SEAKs or PEAKs. Recommended content of the message: • list of purchase item associations, containing – purchase item ID, – time span of validity of contained SEAKs and PEAKs, – list of key records, containing – CID, – SEAK or PEAK, – usage rule info. *This message is network-internal and is not further specified.*

Figure 41 – Interactions for bulk download of service and programme keys

12.2.2.3 Bulk download of purchase information

In cases where the COC is "local" to the SOC, the download of purchase and bundling information from the SOC to the COC(s) providing the broadcast services and content to the users may be implemented as a periodical "bulk download"; see Figure 42.



1	**Get purchase info** Information about purchase items (for example, service bundles) and the items contained in the purchase item (services, schedule items, content items) is fetched from the SOC (its service distribution management) to the COC (its service subscription management); the information how to bundle multiple items into a single purchase item may originate from the service subscription management, but this message still makes sense, because the identifiers of purchase items are assumed to be managed by the service distribution management and synchronized with the identifiers in the service guide. In the case that items that can be subscribed to (service bundles), the subscription options may be indicated to the service subscription management. Recommended content of the message: • time interval for which associations of purchase items and their respective composition and purchase options should be returned. *This message is network-internal and is not further specified.*
2	**OK (purchase info)** The answer to the 'Get purchase info' request contains the purchase info on all purchase items that can currently be sold by the service subscription management. Recommended content of the message: • list of purchase item associations, containing – purchase item ID; – flag whether or not re-keying is used (indicating a subscription); – list of purchase item records, containing (for a service or schedule item or content item): – ID; – name (multi-language); – description (multi-language); – list of purchase option records, containing – purchase option ID; – purchase option description (multi-language); – purchase option price (including currency); The list of purchase item records is not strictly necessary for service subscription management to obtain. However, it can be assumed that for purposes of customer care it will be important for the service subscription management to know what items are included in a purchase item. *This message is network-internal and is not further specified.*

Figure 42 – Interactions for bulk download of purchase information

12.2.2.4 Announcement of purchase items in service guide

The purchase items that relate to the services that are broadcast in the network controlled by the service distribution management are listed in the service guide. In the case that multiple services, schedule items or content items are bundled into a single purchase item, this bundling information shall be part of the service guide. This allows a user to decide which bundle to purchase in order to get access to that particular item.

If not all purchase items are obtainable from a particular COC (through its service subscription management), the information about which purchase items can be obtained from which COC may be included in the service guide.

In the case where a particular purchase item may be obtained via multiple purchase options (for example, one-month subscription, 12-month subscription, renewal option), the service guide should list these purchase options.

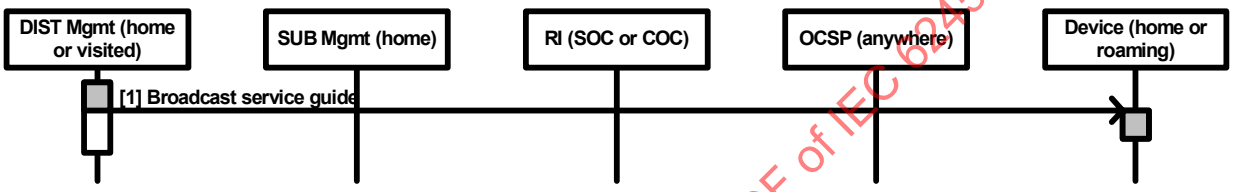
The service guide should include COC-specific price indications for all purchase options.

It is assumed that the service guide will contain availability and pricing information only for "local" COCs. In the case of roaming, the device will have the following options:

a) to ask for this information from its home service subscription management (and execute the purchase via its home COC);

b) to establish a contractual relationship with one of the local COCs who thereby becomes a home COC.

The interaction for the announcement of purchase items in the service guide is shown in Figure 43.



1	**Broadcast service guide** The service guide is assumed to be broadcast, and to contain the purchase-relevant data as specified further down (whereas distribution over a broadcast channel is assumed to be the normal case, the service guide may also be retrievable over an interactivity channel). *The service guide is not within the scope of this standard. However, some purchase-relevant attributes are specified and are assumed to be included in the service guide.*

Figure 43 – Interactions for announcement of purchase items in service guide

12.2.2.5 Pricing inquiry

In the case where the device intends to buy a purchase item that is announced in the service guide, but the service guide

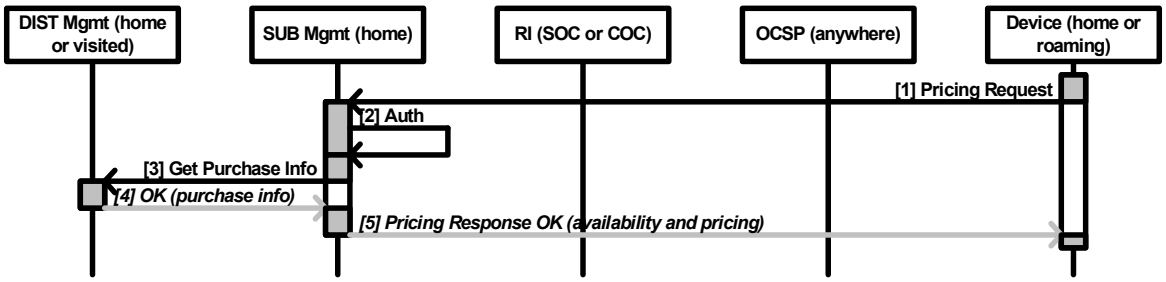
a) contains no information about the home COC of the device; or

b) lacks availability and/or pricing information about the desired purchase item in relation to the home COC

then the device needs to inquire the availability and pricing information from its home COC prior to making a purchase request.

This pricing inquiry is adding an additional interaction between the device and the COC. In order to improve usability and reduce interaction traffic in the "normal" case of home service consumption, the service guide should therefore contain all pricing-relevant information concerning the local COCs.

The interactions for pricing inquiry are specified in Figure 44.



1	**Pricing request** The device may send a pricing request to its home COC. The pricing request refers to a purchase item. In order for the COC to further get the pricing information from the SOC (through its service distribution management), the SOC needs to be identified. The pricing request may include an indication of the user's preferred language. *This message is further specified in 12.2.4.*
2	**Authenticate** Before doing any further processing, the COC authenticates the device and/or user, using some or all of the credentials that are part of the purchase request, or even some credentials that can be supplied by the cellular network, such as MSISDN. *This operation is network-internal and is not further specified.*
3	**Get purchase info** In case the COC doesn't do "bulk download" of purchase and pricing information from the SOC, and typically in the case where the COC has a different geographical location (for example, different country) from the SOC, the COC inquires availability and pricing information from the SOC specifically for the requested purchase item. Recommended content of the message: • purchase item ID; • preferred language. *This message is network-internal and is not further specified.*
4	**OK (purchase info)** In the case where the purchase item requested by the device is valid, the SOC returns its purchase and pricing information (for example, there may be multiple purchasing options with different pricing) back to the COC. Recommended content of the message: • purchase item ID; • flag whether or not re-keying is used (indicating a subscription); • list of purchase item records, containing (for a service or schedule item or content item) – ID (of service or schedule item or content item); – name (user-specified or default language); – description (user-specified or default language); • list of purchase option records, containing – purchase option ID; – purchase option description (user-specified or default language); – purchase option price (including currency). The list of purchase item records is not strictly necessary for COC to obtain. However, it can be assumed that, for the purposes of customer care, it will be important for the COC to know what items are included in a purchase item. *This message is network-internal and is not further specified.*
5	**Pricing response (availability and pricing)** Upon reception of the availability and pricing information from the SOC, the COC may, based on its own logic and pricing rules, modify the pricing and decide whether or not to return a successful pricing response to the device. The successful pricing response lists all pricing options and their prices. The text within the pricing response may be in the user's preferred language, if supported by SOC, otherwise in a default language. *This message is further specified in 12.2.4.*

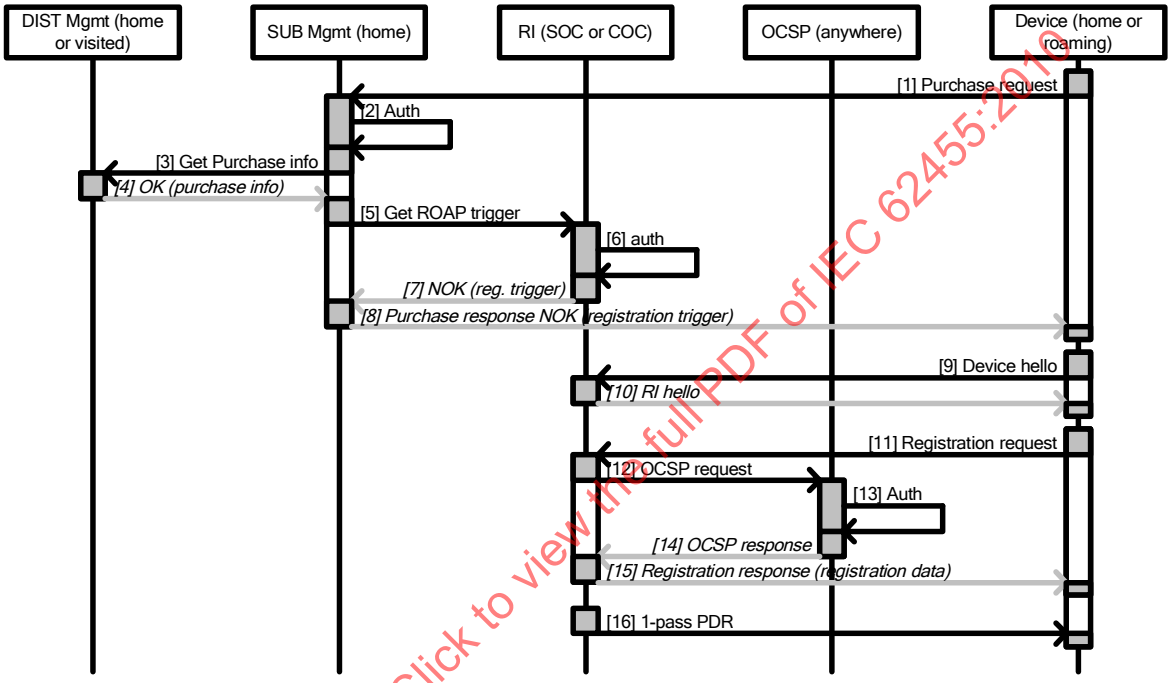
Figure 44 – Interactions for pricing inquiry

12.2.2.6 Unsuccessful purchase

Establishing a contractual relationship (commonly called "subscription" and not to be confused with the subscription of a particular service or service bundle) between a user and a service subscription management is not within the scope of this standard.

Even after a contractual relationship is established, the first purchase might fail, because no device registration has yet been taken place. The following interactions illustrate this scenario, which might be so common that it cannot be treated as an exception.

The interactions for unsuccessful purchase are specified in Figure 45.



1	**Purchase request** In order to initiate a purchase (this can be a one-off purchase of a content item or of a schedule item, or a subscription of a service or service bundle), the device sends a purchase request to its home service subscription management (or one of its home service subscription managements, if there are several of them). Prior to sending the purchase request, the device should have established the availability of the purchase item from the selected service subscription management and its pricing for all purchase options. By sending the purchase request, the device accepts the pricing. *This message is further specified in 12.2.4.*
2	**Authenticate** Before doing any further processing, the service subscription management authenticates the device and/or user, using some or all of the credentials that are part of the purchase request, or even some credentials that can be supplied by the cellular network, such as MSISDN. The device certificate may be returned as a result of this operation. *This operation is network-internal and is not further specified.*

3	**Get purchase info (optional step)** In the case where the service subscription management does not do "bulk download" of purchase and pricing information from the service distribution management, and typically in the case where the service subscription management has a different geographical location (for example, different country) from the service distribution management, the service subscription management inquires availability and pricing information from the service distribution management specifically for the requested purchase item. Recommended content of the message: • purchase item ID; • preferred language. *This operation is network-internal and is not further specified.*
4	**OK (purchase info)** In the case where the purchase item requested by the device is valid, the service distribution management returns its purchase and pricing information (for example, there may be several purchasing options with different pricing) back to the service subscription management. Recommended content of the message: • purchase item ID; • flag whether or not re-keying is used (in case of subscription "yes", otherwise, "no"); • list of purchase item records, containing (for a service or schedule item or content item) – ID (of service or schedule item or content item); – name (user-specified or default language); – description (in user-specified or default language); • list of purchase option records, containing – purchase option ID; – purchase option description (in user-specified or default language); – purchase option price (including currency). The list of purchase item records is not strictly necessary for service subscription management to obtain. However, it can be assumed that, for the purposes of customer care, it will be important for the service subscription management to know what items are included in a purchase item. *This operation is network-internal, and is not further specified.*
5	**Get ROAP trigger** The service subscription management requests a ROAP trigger from the RI. The request may also include (part of) the rights expression that will be included in the RO. It is up to the service subscription management to decide which RI to use (for example, this can be its own RI or the RI of the service distribution management). Recommended content of the message: • RI Device ID; • RI domain ID (optionally used in case the device requests the RO to be valid for a broadcast domain); • purchase item ID (which the RI can use to identify all the keys to pack into the RO); • socID; • socKeyURL; • user-specific rights expression (which the RI puts into the RO, in accordance with rules defined by service distribution management; if defined, these will have to be observed by the device in addition to any post-acquisition usage rules); • current/next flag set to "current" (indicating to the RI that the "current" keys shall be put into the RO). *This operation is network-internal and is not further specified.*

6	**Authenticate** Before doing any further processing, the RI authenticates the device. *This operation is network-internal and is not further specified.*
7	**NOK (reg. trigger)** In case the device is found to have no valid registration (for example, never registered, or the registration expired, or is not trusted anymore), a negative response is sent back to the service subscription management, containing a registration trigger that instructs the device to register. Recommended content of the message: • trigger for device registration (the trigger includes the rights issuer URL to which the registration can be initiated). *This operation is network-internal and is not further specified.*
8	**Purchase response NOK (registration trigger)** The negative response from the RI, including the registration trigger, is forwarded to the device. The ROAP trigger includes the URL of the RI with which the device is supposed to register. *This message is further specified in 12.2.4.*
9	**Device hello** Using the information contained in the trigger, the device initiates the 4-pass device registration. *This is a standard OMA DRM 2.0 operation and is not further specified.*
10	**RI hello** *This is a standard OMA DRM 2.0 operation and is not further specified.*
11	**Registration request** *This is a standard OMA DRM 2.0 operation and is not further specified.*
12	**OCSP request** *This is a standard OMA DRM 2.0 operation and is not further specified.*
13	**Authenticate** Before doing any further processing, the OCSP authenticates the device. *This operation is network-internal and is not further specified.*
14	**OCSP response** *This is a standard OMA DRM 2.0 operation and is not further specified.*

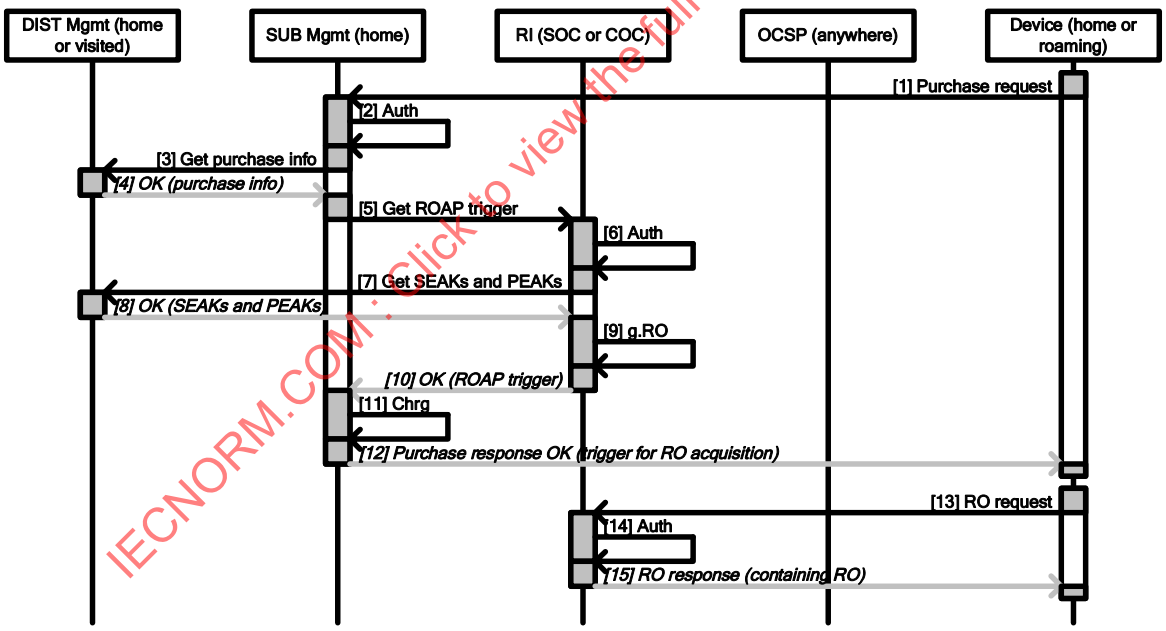
15	**Registration response** As a result of a successful registration, the registration data is securely stored in the device and ready for subsequent use. The device can now re-initiate the purchase transaction. The RI may then include a join domain trigger with the ROAP-RegistrationResponse as an additional MIME part of the payload. This will result in the 2-pass ROAP registration for the OMA DRM 2.0 domain. *This is a standard OMA DRM 2.0 operation and is not further specified.*
16	**1-pass PDR (mixed-mode devices only)** For a mixed-mode device, the RI may decide to send the device_registration_response message, which contains the registration data for the broadcast mode of operation (see 9.3.2.4).

Figure 45 – Interactions for unsuccessful purchase

12.2.2.7 Successful purchase

This sequence is executed if a device is already registered with the RI at the moment when it initiates the purchase transaction. It is also assumed that all authentication steps are carried out successfully, and the purchase can be successfully charged.

The Interactions for successful purchase are specified in Figure 46.



1	**Purchase request** In order to initiate a purchase (this can be a one-off purchase of a content item or of a schedule item, or a subscription of a service or service bundle), the device sends a purchase request to its home service subscription management (or one of its home service subscription management, if there are several of them). Prior to sending the purchase request, the device should have established the availability of the purchase item from the selected service subscription management, and its pricing for all purchase options. By sending the purchase request, the device accepts the pricing. *This message is further specified in 12.2.4.*

2	**Authenticate** Before doing any further processing, the service subscription management authenticates the device and/or user, using some or all of the credentials that are part of the purchase request, or even some credentials that can be supplied by the cellular network, such as MSISDN. The device certificate may be returned as a result of this operation. *This operation is network-internal and is not further specified.*
3	**Get purchase info (optional step)** In the case where the service subscription management does not do "bulk download" of purchase and pricing information from the service distribution management and typically in the case where the service subscription management has a different geographical location (for example, different country) from the service distribution management, the service subscription management requests availability and pricing information from the service distribution management specifically for the requested purchase item. Recommended content of the message: • purchase item ID; • preferred language. *This operation is network-internal and is not further specified.*
4	**OK (purchase info)** In the case where purchase item requested by the device is valid, the service distribution management returns its purchase and pricing information (for example, there may be multiple purchasing options with different pricing) back to the service subscription management. Recommended content of the message: • purchase item ID; • flag whether or not re-keying is used (in case of subscription, "yes", otherwise, "no"); • list of purchase item records, containing (for a service or schedule item or content item) – ID (of service or schedule item or content item); – name (user-specified or default language); – description (user-specified or default language); • list of purchase option records, containing – purchase option ID; – purchase option description (in user-specified or default language); – purchase option price (including currency). The list of purchase item records is not strictly necessary for service subscription management to obtain. However, it can be assumed that for purposes of customer care it will be important for the service subscription management to know what items are included in a purchase item. *This operation is network-internal and is not further specified.*

5	**Get ROAP trigger** The service subscription management requests a ROAP trigger from the RI. If the request includes any rights expression, they will be included in the RO. It is up to the service subscription management to decide which RI to use (for example, this can be the own RI of the service subscription management, or the RI of the service distribution management). Recommended content of the message: • RI Device ID; • RI domain ID (optionally used in case the device requests the RO to be valid for a broadcast domain); • purchase item ID (which the RI can use to identify all the keys to pack into the RO); • socID; • socKeyURL; • user-specific rights expression (which the RI puts into the RO, in accordance with rules defined by service distribution management; if defined, these will have to be observed by the device in addition to any post-acquisition usage rules); • current/next flag set to "current" (indicating to the RI that the "current" keys shall be put into the RO). *This operation is network-internal and is not further specified.*
6	**Authenticate** Before doing any further processing, the RI authenticates the device. In this example of a purchase sequence, the device is found to have a valid registration (RI context). *This operation is network-internal and is not further specified.*
7	**Get SEAKs and PEAKs (optional step)** In the case that there is no "bulk download" of associations between purchase items and the corresponding keys (service keys, programme keys) from the service distribution management to the RI, the RI requests the necessary keys by sending the purchase item identification to the service distribution management. Recommended content of the message: • purchase item ID; • current/next flag set to "current" (indicated that the "current" keys are requested). *This operation is network-internal and is not further specified.*

8	**OK (SEAKs and PEAKs)** The service and programme keys corresponding to the purchase item as specified in the request are returned by the service distribution management to the RI. There may be some information regarding usage rules specified, which influence the generation of the rights expression by the RI. Recommended content of the message: • purchase item ID; • time span of validity of contained SEAKs or PEAK; • list of key records, containing – CID; – SEAK or PEAK; • usage rule info. *This operation is network-internal and is not further specified.*
9	**Generate RO** The RI generates the RO containing the desired keys (for example, multiple SEAKs in case of a subscription to a service bundle, or a PEAK in case of pay-per-view of a service). In the case where the service subscription management has specified some particular usage rights, these are included in the rights expression of the RO, under consideration of the usage rules that may have been specified by the service distribution management. *This operation is network-internal and is not further specified.*
10	**OK (ROAP trigger)** In the case of success, the RI sends an OK message back to the service subscription management, containing also the ROAP trigger that the device may use to request the prepared RO. Recommended content of the message: • trigger for the rights object acquisition (the trigger includes the rights issuer URL from which the RO can be acquired by the device). The trigger may be omitted for mixed-mode devices (see 12.3). The device certificate may also be returned as a result of this operation. *This operation is network-internal and is not further specified.*
11	**Charge** Before sending the RO acquisition trigger back to the device, the service subscription management may charge the user (if consumption-based charging based on the consumption of tokens is used, the user is not charged when the "purchase request" is made, but when tokens are requested). How charging is done (for example, credit card, direct bank debit, generation of charging record to be included in a periodical invoice) is completely up to the implementation and to the contract between the user and the service subscription management. *This operation is network-internal and is not further specified.*

12	**Purchase response OK (trigger for RO acquisition)** As part of the positive response to the purchase request, the trigger for RO acquisition is sent to the device. The ROAP trigger includes the URL of the RI that the device can use to retrieve the prepared RO. Alternatively, if the device is a mixed-mode device the RI may decide to deliver a BCRO over a broadcast channel. In this case, the ROAP trigger will not be delivered to the device and the flow continues with the delivery of a BCRO (see 12.3). If the request was for a domain RO, and the device has not yet joined the domain, the RI may include a join domain trigger as an additional MIME part of the response payload. The join domain operation is not described in this sequence. *This message is further specified in 12.2.4.*
13	**RO request** Using the information contained in the ROAP trigger, the device initiates the 2-pass ROAP. *This is a standard OMA DRM 2.0 operation and is not further specified.*
14	**Authenticate** Before doing any further processing, the RI authenticates the device and/or user. *This operation is network-internal and is not further specified.*
15	**RO response** As a result of a successful RO acquisition, the RO is available in the device. *This is a standard OMA DRM 2.0 operation and is not further specified.*

Figure 46 – Interactions for successful purchase

12.2.2.8 Subscription RO renewal and asynchronous charging

The ROs for subscriptions will have to be periodically renewed. It can be expected that the lifetime of a subscription RO will be on the order of 1 day to 1 month and no longer than the subscription period. A shorter lifetime means higher security at the expense of more processing and bandwidth usage.

Renewing a subscription RO can be understood as "re-keying of the service key". The purpose is to provide the device with the "next service key" for all services that a device or user is already subscribed to, and may already have paid for. The need for authentication by the service subscription management makes the RO renewal request very similar in nature to the normal purchase request, and the data flows are largely identical.

The device may initiate RO renewal any time during the lifetime of the "current" RO. The lifetime of the RO is signalled in form of a "date-time" restriction to the "access" permission of the RO. In order to avoid all devices to renew their ROs at the same time, the following random delay mechanism shall be used to spread renewal over the whole renewal period.

$T1$ = the point in time when all of the SEAKs in the current RO first became active.

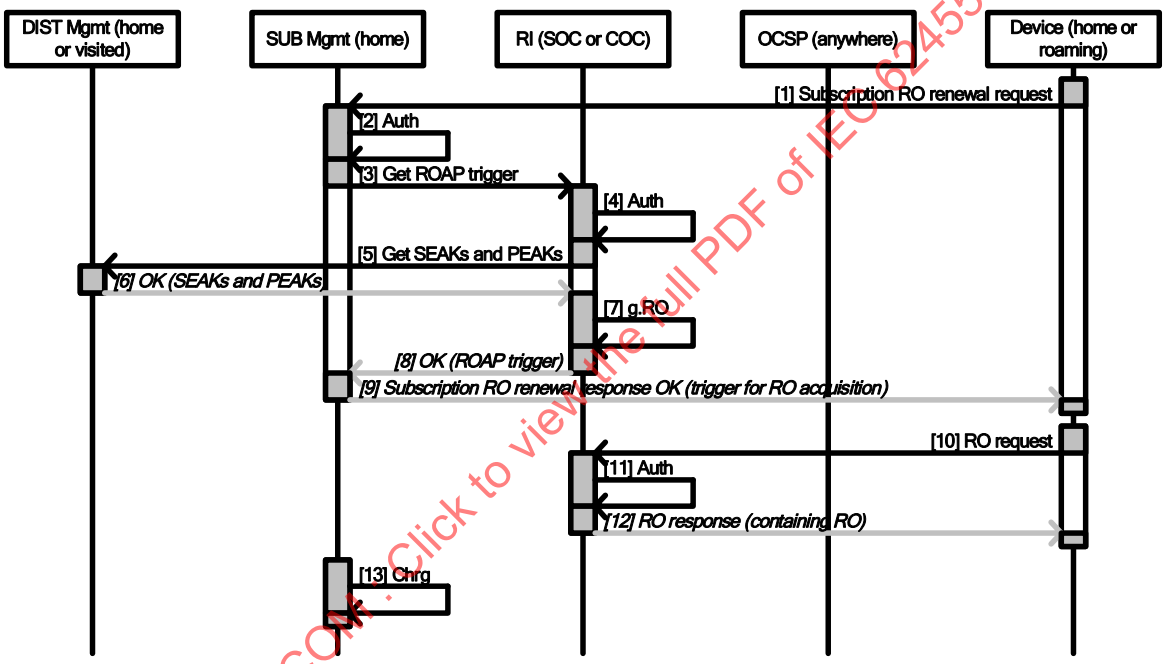
$T2$ = the point in time when all of the SEAKs in the current RO are due to expire.

DT = a device shall request the next RO within DT of the current RO's expiry (implementation-dependent and big enough to ensure that the device gets the next RO timely). It is expected that DT is signalled in the service guide.

$T1$ and $T2$ define the time interval in the "date-time" restriction in the "access" permission in the RO. The device shall request the next RO at a random point in time T, where $T1$ <= T <= $T2$-DT.

In case of open-ended subscriptions, it is expected that the service subscription management will periodically charge the user. If this is the case, the charging operation should be completely separated from the RO renewal, and may continue until cancellation of the subscription, whether the device renews the related RO or not.

The interactions for subscription RO renewal and asynchronous charging are specified in Figure 47.



1	**Subscription RO renewal request** The device sends the renewal request to the same service subscription management from where it originally purchased the subscription. This allows the service subscription management to check that the renewal request indeed corresponds to a valid subscription. In order to signal to the service subscription management which RO to return, the renewal request contains the expiry time of the current RO. If no expiry time is given, then the service subscription management will assume that the current RO has been lost and treat the request as a replacement request for the current RO. *This message is further specified in 12.2.4.*
2	**Authenticate** Before doing any further processing, the service subscription management authenticates the device and/or user, using some or all of the credentials that are part of the purchase request, or even some credentials that can be supplied by the cellular network, such as MSISDN. The device certificate may be returned as a result of this operation. *This operation is network-internal and is not further specified.*

3	**Get ROAP Trigger** The service subscription management requests a ROAP trigger from the RI. The request may also include (part of) the rights expression that will be included in the RO. The service subscription management will use the same RI as for the original purchase request. Recommended content of the message: • RI device ID; • RI domain ID (optionally used in case the device requests the RO to be valid for a broadcast domain); • purchase item ID (which the RI can use to identify all the keys to pack into the RO); • socID; • socKeyURL; • user-specific rights expression (which the RI puts into the RO, in accordance with rules defined by service distribution management; if defined, these will have to be observed by the device in addition to any post-acquisition usage rules); • current/next flag set to "next" (indicating to the RI that the "next" keys shall be put into the RO). *This operation is network-internal and is not further specified.*
4	**Authenticate** Before doing any further processing, the RI authenticates the device. *This operation is network-internal and is not further specified.*
5	**Get SEAKs and PEAKs (optional step)** In the case that there is no "bulk download" of associations between purchase items and the corresponding keys (SEAK, PEAK) from the service distribution management to the RI, the RI requests the necessary SEAKs and PEAKs by sending the purchase item identification to the service distribution management. Importantly, the service subscription management will specify whether the "current" or "next" keys are desired. Recommended content of the message: • purchase item ID; • current/next flag set to "current" (indicated that the "current" keys are requested). *This operation is network-internal and is not further specified.*

6	**OK (SEAKs and PEAKs)** The SEAKs or PEAKs corresponding to the purchase item as specified in the request are returned by the service distribution management to the RI. There may be some information regarding the usage rules specified which influence the generation of the rights expression by the RI. Recommended content of the message: • purchase item ID; • time span of validity of contained SEAKs or PEAK; • list of key records, containing – CID; – SEAK or PEAK; • usage rule info. *This operation is network-internal and is not further specified.*
7	**Generate RO** The RI generates the RO containing the desired keys. In the case that the service subscription management has specified some particular usage rights, these are included in the rights expression of the RO, under consideration of the usage rules that may have been specified by the service distribution management. *This operation is network-internal and is not further specified.*
8	**OK (ROAP trigger)** In the case of success, the RI sends an OK message back to the service subscription management, containing also the ROAP trigger that the device may use to request the prepared RO. The device certificate may also be returned as a result of this operation. Recommended content of the message: • trigger for the rights object acquisition (the trigger includes the rights issuer URL from which the RO can be acquired by the device). *This operation is network-internal and is not further specified.*
9	**Subscription RO renewal response OK (trigger for RO acquisition)** As part of the positive response to the purchase request, the trigger for the RO acquisition is sent to the device. The ROAP trigger includes the URL of the RI that the device can use to retrieve the prepared RO. Alternatively, if the device is a mixed-mode device the RI may decide to deliver a BCRO over a broadcast channel. In this case, the ROAP trigger will not be delivered to the device and the flow continues with the delivery of a BCRO (see 12.3). *This message is further specified in 12.2.4.*
10	**RO Request** Using the information contained in the ROAP trigger, the device initiates the 2-pass ROAP. *This is a standard OMA DRM 2.0 operation and is not further specified.*

11	**Authenticate** Before doing any further processing, the RI authenticates the device and/or user. *This operation is network-internal and is not further specified.*
12	**RO response** As a result of a successful RO acquisition, the RO is available in the device. *This is a standard OMA DRM 2.0 operation and is not further specified.*
13	**Charge** How charging is done (for example, credit card, direct bank debit, generation of charging record to be included in a periodical invoice) is completely up to the implementation and to the contract between the user and service subscription management. *This operation is network-internal and is not further specified.*

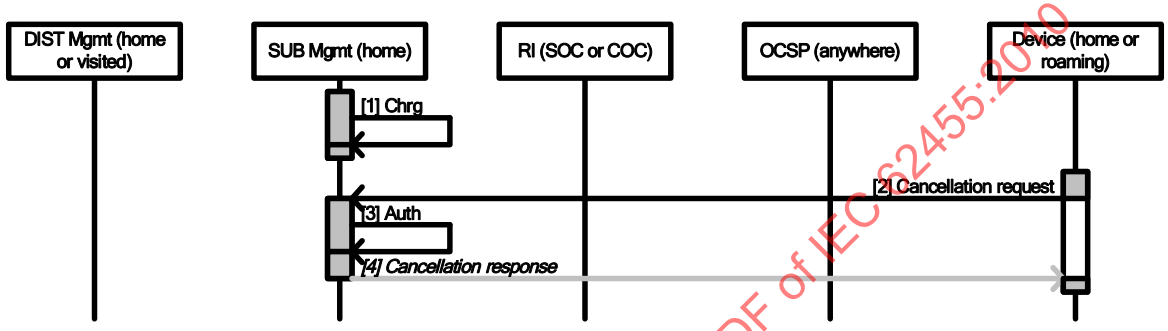
Figure 47 – Interactions for subscription RO renewal and asynchronous charging

12.2.2.9 Asynchronous charging and cancellation of open-ended subscriptions

In the case of open-ended subscriptions, the user is charged asynchronously from time to time, irrespective of any RO renewals. Typically, such asynchronous charging could happen on a monthly basis.

Open-ended subscriptions are valid until they are cancelled by the user. Depending on the contract, they may also have to be cancelled (and renewed by issuing a new purchase request) when the price per subscription period changes.

The interactions for asynchronous charging and cancellation of open-ended subscriptions are specified in Figure 48.



1	**Charge** How charging is done (for example, credit card, direct bank debit, generation of charging record to be included in a periodical invoice) is completely up to the implementation and to the contract between the user and service subscription management. *This operation is network-internal and is not further specified.*
2	**Cancellation request** The device sends the cancellation request to the same service subscription management from where it originally purchased the subscription. If the cancellation is received only after the device has already retrieved the ROs pertaining to the next subscription period, cancellation may become effective at the end of the current or at the end of the next subscription period. The cancellation may or may not have an immediate effect. If the user has already paid for the current subscription period, the subscription RO renewal is expected to succeed until the current subscription period is over, and fails thereafter. *This message is further specified in 12.2.4.*
3	**Authenticate** Before doing any further processing, the service subscription management authenticates the device and/or user, using some or all of the credentials that are part of the purchase request, or even some credentials that can be supplied by the cellular network, such as MSISDN. The device certificate may be returned as a result of this operation. *This operation is network-internal and is not further specified.*
4	**Cancellation response** The cancellation response includes a text that tells the user until when the cancelled service can still be received. This means that the device should continue renewing ROs until the renewal fails. *This message is further specified in 12.2.4.*

Figure 48 – Interactions for asynchronous charging and cancellation of open-ended subscriptions

12.2.2.10 Purchase of tokens for consumption-based charging

The following sequence shows the interactions needed for the purchase of tokens that can be used for the token consumption-based pre- and post-paid models.

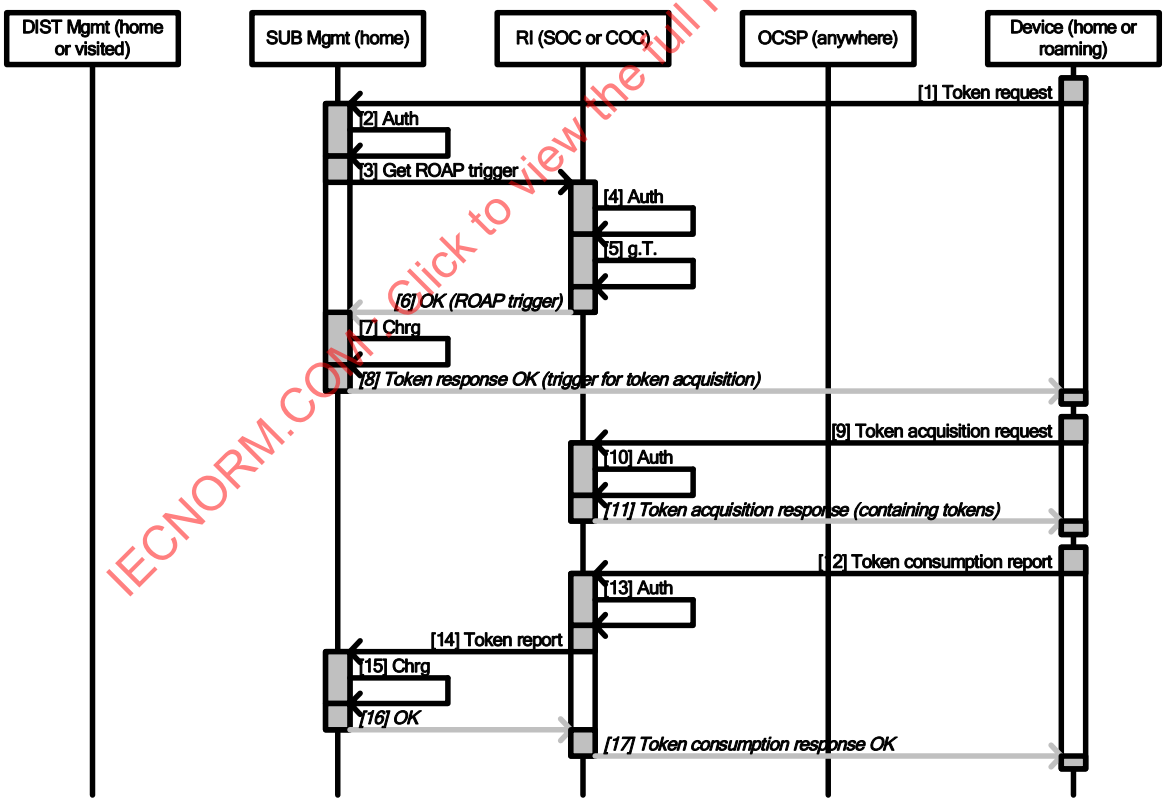
The purchase of tokens is fully separated from the consumption book-keeping and independent of the existence of particular purchase items. However, it may depend on the contract a user has with the subscription management whether or not that particular user can purchase tokens and whether to use pre- or post-paid mode.

Concerning the delivery of tokens, the interactions for pre- and post-paid mode are identical, but the charging step bears different semantics:

- in the pre-paid case, the tokens that are delivered as a result of the purchase transaction are charged immediately;
- in the post-paid case, the charging is done after the token usage has been reported by the device.

The mechanisms for acquiring the token amount and reporting the token usage using the interactivity channel are specified in OMA XBS. These mechanisms may later on become incorporated in future versions of OMA-ERP-DRM-V2_0.

The interactions for acquisition and charging of tokens are specified in Figure 49.



1	**Token request** Based on an internal logic, which is not in the scope of this standard, the terminal or user decides to purchase additional tokens from the subscription management and issues a token request. The attributes of the token request are: • the type of charging (pre-paid or post-paid); • the requested number of tokens. In pre-paid mode, these will be charged. In post-paid mode, they indicate the requested credit limit (the actual number of tokens that will be sent to the device is up to the subscription management). *This message is further specified in 12.2.4.*
2	**Authenticate** Before doing any further processing, the subscription management authenticates the terminal and/or user, using some or all of the credentials that are part of the purchase request, or even some credentials that can be supplied by the cellular network, such as MSISDN. The device certificate may be returned as a result of this operation. *This operation is network-internal and is not further specified.*
3	**Get ROAP trigger** The subscription management requests a ROAP trigger from the RI. The request may also include (part of) the rights expression that will be included in the RO. The subscription management will use the same RI as for the original purchase request. Recommended content of the message: • RI Device ID; • number of tokens; • in the case of post-paid, an indication that a token consumption report is requested for this token delivery and the URL to which the RI should send the token consumption report. *This operation is network-internal and is not further specified.*
4	**Authenticate** Before doing any further processing, the RI authenticates the terminal and/or user. *This operation is network-internal and is not further specified.*
5	**Generate token** The RI generates the desired amount of tokens. *This operation is network-internal and is not further specified.*

6	**OK (ROAP trigger)** In the case of success, the RI sends an OK message back to the subscription management, containing also the ROAP trigger that the terminal may use to request the prepared tokens. Recommended content of the message: • trigger for the rights object acquisition (the trigger includes the rights issuer URL from which the token RO can be acquired by the terminal); • the device certificate may also be returned as a result of this operation. *This operation is network-internal and is not further specified.*
7	**Charge (pre-paid only)** How charging is done (for example, credit card, direct bank debit, generation of charging record to be included in a periodical invoice) is completely up to the implementation and to the contract between the user and subscription management. This step is executed only if pre-paid charging is selected. *This operation is network-internal and is not further specified.*
8	**Token response OK** As part of the positive response to the token request, the trigger for token acquisition is sent to the terminal. The ROAP trigger includes the URL of the RI that the terminal can use to retrieve the prepared token. *This message is further specified in 12.2.4.*
9	**Token acquisition request** Using the information contained in the ROAP trigger, the terminal initiates the 2-pass ROAP. *This is a standard OMA DRM 2.0 operation and is not further specified.*
10	**Authenticate** Before doing any further processing, the RI authenticates the terminal and/or user. *This operation is network-internal and is not further specified.*
11	**Token acquisition response** As a result of a successful token acquisition, the token is available in the terminal. *This is a standard OMA DRM 2.0 operation and is not further specified.*
12	**Token consumption report (post-paid only)** If post-paid charging was selected, the device reports the token consumption to the RI. The rest of this sequence applies only to the post-paid charging case. *This is a standard OMA DRM 2.0 operation and is not further specified.*
13	**Authenticate** Before doing any further processing, the RI authenticates the terminal and/or user. *This operation is network-internal and is not further specified.*

14	**Token report** After having validated it, the RI forwards the token report to the subscription management. *This operation is network-internal and is not further specified.*
15	**Charge** The subscription management charges the user based on the amount of tokens consumed. *This operation is network-internal and is not further specified.*
16	**OK** *This operation is network-internal and is not further specified.*
17	**Token consumption report OK** *This is a standard OMA DRM 2.0 operation and is not further specified.*

Figure 49 – Interactions for acquisition and charging of tokens

12.2.3 Protocol

12.2.3.1 General

The service provider shall support HTTP POST, which may be used for purchase requests over an interactivity channel.

The service provider may support HTTPS POST, which may be used for purchase requests over an interactivity channel.

The service provider shall use either HTTP POST or HTTPS POST for purchase requests over an interactivity channel.

The device shall support both HTTP POST and HTTPS POST for purchase requests over an interactivity channel.

The device needs to know the URL for HTTP or HTTPS sessions. It is expected that this is supported by information contained in the service guide.

12.2.3.2 HTTP headers

Request messages shall be sent as HTTP content of type "application/xml". Responses shall always be sent as part of the "200 OK" response to the original request. The content type shall be "application/xml" if the response has only one payload, or "multipart/mixed" in the case of multiple payloads (for example, if the response includes one or more ROAP triggers). In the latter case, the content type of a single payload of the multipart content shall be "application/xml" for the messages defined here and "application/vnd.oma.drm.roap-trigger+xml" for the ROAP triggers.

12.2.3.3 Signatures

All request messages shall be signed with the private device key, using the RSASSA-PSS algorithm as specified in IETF RFC 3447 (see also Clause A.10). The input to the signature operation shall be the XML payload of the request without the "signature" element, canonicalised according to the exclusive XML canonicalisation algorithm; see OMA-TS-DRM-DRM-V2_0.

12.2.4 XML schemas for request and response messages

12.2.4.1 Basic types

12.2.4.1.1 General

This subclause contains the XML schema fragment definitions for the elementary data types used in the subsequent message definitions.

12.2.4.1.2 User data type

The user data includes the user's identity known to the customer operator centre that will be used for billing and, optionally, the user's preferred language (specified as a two-letter code as defined in ISO 639-1:2002). Below is the XML schema fragment for the UserDataType.

```
<xs:complexType name="userDataType">
  <xs:sequence>
    <xs:element name="userID" type="xs:string"/>
    <xs:element name="lang" type="xs:language" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

12.2.4.1.3 Device data type

The device data includes the device identification and the device capabilities. Device capabilities are optional and, if they are omitted, the subscription management may assume the capabilities announced in a previous request. However, if the device has never announced its capabilities to the subscription management, the capabilities shall be included in the request.

The detailed device data are:

- the device ID known to the subscriber management. For mixed-mode devices, the content of this field shall be the UDN of the device;
- the device ID known to the right issuer, specified in OMA-ERP-DRM-V2_0;
- the broadcast bearers supported by the device (currently defined identifiers: "dvbh", "mbms", "bcmcs");
- the service protection protocols supported by the device (currently defined identifiers: "ipsec", "srtp", "ismacryp");
- the device type ("interactive" for interactive-only devices, "broadcast" for broadcast-only devices, "mixed" for mixed-mode devices).

```
<xs:complexType name="deviceDataType">
  <xs:sequence>
    <xs:element name="deviceID" type="xs:token"/>
    <xs:element name="riDeviceID" type="xs:token"/>
    <xs:element name="bcastBearer" type="xs:token" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="serviceProtection" type="xs:token" minOccurs="0"
                                         maxOccurs="unbounded"/>
    <xs:element name="deviceType" type="xs:token" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

12.2.4.1.4 Domain type

If the service guide indicates that the purchase is available for a domain, the user is allowed to select such an option by including the domain ID in the purchase request. For mixed-mode devices, the domain data also specifies whether the domain is an OMA DRM 2.0 domain ("omadrm2") or a broadcast domain ("broadcast").

```
  <xs:complexType name="domainType">
   <xs:sequence>
     <xs:element name="domainID" type="xs:token"/>
     <xs:element name="domainType" type="xs:token" minOccurs="0"/>
   </xs:sequence>
 </xs:complexType>
```

12.2.4.1.5 ServiceOperatorCentreType

The service operation centre data contains the following information.

NOTE There are two XML elements describing the service operation centre. One XML element, the ServiceOperationCentreType is used in ESG signalling while this XML element of the type ServiceOperatorCentreType is used in request and response messages.

socID: Globally coordinated ID of the service operation centre.

socInfoURL: The URL through which the SubMgmt can retrieve purchase information from the SOC.

socKeyURL: The URL from which an RI can receive service and programme keys.

riID: Globally coordinated ID of the rights issuer.

riURL: The rights Issuer URL, from which the SubMgmt can retrieve the ROAP triggers that will be delivered to the device.

riProxyURL: The RI proxy URL, from which the broadcast of BCROs in a foreign network can be requested (see B.8.3). If this parameter is received via the service guide, it is mandatory to include it in the SOC data.

If SOC information is included in one of the requests, the socID is always MANDATORY. riProxyURL is always OPTIONAL. The other elements may or may not be needed, depending on the particular message, as illustrated in Table 106.

Table 106 – Definition of mandatory SOC attributes in request/response messages

Message\Element	socID	socInfoURL	socKeyURL	riID	riURL	riProxyURL
Pricing request	M	M				
Purchase request	M	M	M	M	M	
Renewal request	M		M	M	M	
Cancel request	M					
Token request	M			M	M	
<Any> response	M					
NOTE "M" means that the element is MANDATORY for the particular request.						

Below is the XML schema fragment for the ServiceOperatorCentreType.

```
  <xs:complexType name="serviceOperatorCentreType">
   <xs:sequence>
     <xs:element name="socID" type="xs:token"/>
     <xs:element name="socInfoURL" type="xs:anyURI" minOccurs="0"/>
     <xs:element name="socKeyURL" type="xs:anyURI" minOccurs="0"/>
     <xs:element name="riID" type="xs:token" minOccurs="0"/>
     <xs:element name="riURL" type="xs:anyURI" minOccurs="0"/>
     <xs:element name="riProxyURL" type="xs:anyURI" minOccurs="0"/>
   </xs:sequence>
 </xs:complexType>
```

12.2.4.1.6 PriceType

The price information contains the price as a decimal value and an optional currency qualifier (specified as defined in ISO 4217), as illustrated by the following XML schema fragment.

```
<xs:complexType name="priceType">
  <xs:simpleContent>
    <xs:extension base="xs:decimal">
      <xs:attribute name="currency" type="xs:token" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

12.2.4.1.7 Purchase item type

The purchase item data contains the information needed to identify the user's purchase.

ItemID: the purchase_item_id received from the service guide.

PurchaseOption: an identifier of the particular type of purchase, for example, open-ended subscription or one-time subscription, a received from the service guide.

Price: the price of the item known to the user (optional).

Version: the version number of this purchase item, as indicated in the service guide.

Below is the XML schema fragment for the PurchaseItemType.

```
<xs:complexType name="purchaseItemType">
  <xs:sequence>
    <xs:element name="itemID" type="xs:token"/>
    <xs:element name="version" type="xs:nonNegativeInteger" minOccurs="0"/>
    <xs:element name="purchaseOption" type="xs:token" minOccurs="0"/>
    <xs:element name="price" type="s18:priceType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

12.2.4.1.8 Request type

Each request message extends the base request type. The request type has one "version" attribute that indicates the interface version used for this communication.

```
<xs:complexType name="requestType">
  <xs:attribute name="interfaceVersion" type="xs:nonNegativeInteger" use="required"/>
</xs:complexType>
```

12.2.4.1.9 Response type

Each response message extends the base response type. The response type has a "status" attribute, which is either success or error, a "reason code" attribute that indicates the cause of error among the ones listed in Table 107 and an optional free-form error text.

```
<xs:complexType name="responseType">
  <xs:sequence>
    <xs:element name="message" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="status" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="success"/>
        <xs:enumeration value="error"/>
      </xs:restriction>
    </xs:simpleType>
```

```
    </xs:attribute>
    <xs:attribute name="reasonCode" type="xs:nonNegativeInteger" use="optional"/>
  </xs:complexType>
```

12.2.4.2 Error codes

Table 107 lists all the possible reasonCode values for error case and their applicability to each transaction.

Table 107 – Occurrence of error codes in response messages

Code	Error situation	Pricing	Purchase	Renewal	Cancel	Token
0	**Authentication failed** This code indicates that the service subscription management was unable to authenticate the user or the device, which may be due to the fact that the user or the device is not registered with the service subscription management. In this case, the user may contact the service subscription management and establish a contract, or get the credentials in place that are used for authentication.	X	X	X	X	X
1	**Purchase item unknown** This code indicates that the requested purchase item is unknown. This can happen, for example, if the device has a cached service guide with old information. In this case, the user may re-acquire the service guide.	X	X			
2	**Device not authorized** This code indicates that the device is not authorized to get ROs from the RI, for example, because the device certificate was revoked. In this case, the user may contact the service subscription management operator.		X	X		X
3	**Device not registered** This code indicates that the device is not registered with the RI that is used for the transaction. When this code is sent, the response message includes a registration trigger that allows the device to register. In this case, the device may automatically perform the registration, and, if the registration is successful, re-initiate the original transaction.		X	X		X
4	**Server error** This code indicates that there was a server error, such as a problem connecting to a remote back-end system. In such a case, the transaction may succeed if it is re-initiated later.	X	X	X	X	X
5	**Device error** This code indicates that there has been a device malfunction, such as a mal-formed XML request. In such a case, the transaction may or may not (for example, if there is an interoperability problem) succeed if it is re-initiated later.	X	X	X	X	X

Code	Error situation	Pricing	Purchase	Renewal	Cancel	Token
6	**Charging error** This code indicates that the charging step failed (for example, agreed credit limit reached, account blocked) and therefore the requested RO cannot be provided. The user may in such a case contact the service subscription management operator.		X			X
7	**No subscription** This code indicates that there has never been a subscription for this purchase item, or that the subscription for this purchase item has terminated. The user may in such a case issue a purchase request for a new subscription.			X	X	
8	**Operation not permitted** This code indicates that the operation that the device attempted to perform is not permitted under the contract between service subscription management and user. The user may in this case contact service subscription management operator and change the contract.	X	X	X	X	X
9	**Unsupported version** This code indicates that the version number specified in the request message is not supported by the network. In this case, the user may contact the service subscription management operator.	X	X	X	X	X
10	**Signature error** This code indicates that the validation of the message signature failed. In this case, the user may contact the service subscription management operator.	X	X	X	X	X
11	**Domain error** This code indicates that the device has requested a purchase for a domain to which it does not belong. When this code is sent, the response message includes a "join domain" trigger that allows the device to join the domain if it has the necessary permissions. In this case, the device may automatically perform the join domain procedure, and, if the operation is successful, re-initiate the original transaction.		X	X		
12	**Purchase item version mismatch** This code indicates that the purchase item requested by the device has an older version number compared to the one stored in the COC. In this case, the device should update its service guide and retry the purchase.		X			
NOTE "X" means that only the particular request may result in the corresponding error code.						

12.2.4.3 Pricing request

12.2.4.3.1 General

The pricing request includes the user, device and SOC data specified above, and a list of the identifiers of the purchase items for which the user is requesting the price details.

12.2.4.3.2 Schema

```
 <xs:element name="pricingRequest" type="c18:pricingRequestType"/>
 <xs:complexType name="pricingRequestType">
   <xs:complexContent>
     <xs:extension base="c18:requestType">
       <xs:sequence>
         <xs:element name="user" type="c18:userDataType"/>
         <xs:element name="device" type="c18:deviceDataType"/>
         <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
         <xs:element name="purchaseItemList">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="purchaseItem" type="c18:purchaseItemType"
maxOccurs="unbounded"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
         <xs:element name="signature" type="xs:base64Binary"/>
       </xs:sequence>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
```

12.2.4.3.3 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<pricingRequest xmlns="http://www.18crypt.com/2005/XMLSchema"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance interfaceVersion="1">
   <user>
      <userID>24403123456</userID>
      <lang>en</lang>
   </user>
   <device>
      <deviceID>0044005817853</deviceID>
      <riDeviceID>1234567890</riDeviceID>
      <bcastBearer>dvbh</bcastBearer>
      <serviceProtection>ipsec</serviceProtection>
      <deviceType>interactive</deviceType>
   </device>
   <serviceOperatorCentre>
      <socID>12345</socID>
      <socInfoURL>http://www.soc.com/info</socInfoURL>
   </serviceOperatorCentre>
   <purchaseItemList>
      <purchaseItem>
         <itemID>5432</itemID>
      </purchaseItem>
   </purchaseItemList>
   <signature>f7jwx3rvEPO0vKtMup4NbeVu9kn=</signature>
</pricingRequest>
```

12.2.4.4 Pricing response

12.2.4.4.1 General

The pricing response contains a global status code ("success" or "error") and an optional global failure code in case the transaction failed completely. The response includes a list of purchase items and for each of them,

- in the case of success, the purchase options and their price;
- otherwise the item-specific error code.

12.2.4.4.2 XML Schema

```
<xs:element name="pricingResponse" type="c18:pricingResponseType"/>
<xs:complexType name="pricingResponseType">
  <xs:complexContent>
    <xs:extension base="c18:responseType">
      <xs:sequence minOccurs="0">
        <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
        <xs:element name="purchaseItemList">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="purchaseItem" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:choice>
                    <xs:element name="purchaseOptionList">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="purchaseOption" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="optionID" type="xs:token"/>
                                <xs:element name="desc" type="xs:string"/>
                                <xs:element name="price" type="c18:priceType"/>
                              </xs:sequence>
                              <xs:attribute name="type" use="required">
                                <xs:simpleType>
                                  <xs:restriction base="xs:token">
                                    <xs:enumeration value="one-time"/>
                                    <xs:enumeration value="continuous"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:attribute>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="reasonCode" type="xs:negativeInteger"/>
                  </xs:choice>
                  <xs:attribute name="itemID" type="xs:token"/>
                  <xs:attribute name="version" type="xs:nonNegativeInteger"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```


12.2.4.4.3 Example: successful pricing response

The successful pricing response contains for each individual purchase item the same purchase information that is available through the service guide, though limited to the service subscription management to which the pricing request has been addressed, and containing only the availability and pricing-related information.

```
<?xml version="1.0" encoding="UTF-8"?>
<pricingResponse status="success">
   <serviceOperatorCentre>
      <socID>12345</socID>
   </serviceOperatorCentre>
   <purchaseItemList>
      <purchaseItem itemID="5432" version="1234345">
         <purchaseOptionList>
            <purchaseOption type="one-time">
               <optionID>A</optionID>
               <desc>one-month subscription</desc>
               <price currency="EUR">5.00</price>
            </purchaseOption>
            <purchaseOption type="continuous">
               <optionID>B</optionID>
               <desc>monthly subscription, renewable until cancellation</desc>
               <price currency="EUR">5.00</price>
```

```
            </purchaseOption>
         </purchaseOptionList>
      </purchaseItem>
   </purchaseItemList>
</pricingResponse>
```

12.2.4.5 Purchase request

12.2.4.5.1 General

The purchase request includes the user, device and SOC data specified above, and a list of the identifiers of items the user wants to purchase, including a purchase option and the price known to the user. The purchase can be requested for a domain, in which case optional domain data is specified.

12.2.4.5.2 Schema

```
 <xs:element name="purchaseRequest" type="c18:purchaseRequestType"/>
 <xs:complexType name="purchaseRequestType">
   <xs:complexContent>
     <xs:extension base="c18:requestType">
       <xs:sequence>
         <xs:element name="user" type="c18:userDataType"/>
         <xs:element name="device" type="c18:deviceDataType"/>
         <xs:element name="domain" minOccurs="0">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="domainID"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
         <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
         <xs:element name="purchaseItemList">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="purchaseItem" type="c18:purchaseItemType"
maxOccurs="unbounded"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
         <xs:element name="signature" type="xs:base64Binary"/>
       </xs:sequence>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
```

12.2.4.5.3 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseRequest interfaceVersion="1">
   <user>
      <userID>24403123456</userID>
      <lang>en</lang>
   </user>
   <device>
      <deviceID>0044005817853</deviceID>
      <riDeviceID>1234567890</riDeviceID>
      <bcastBearer>dvbh</bcastBearer>
      <serviceProtection>ipsec</serviceProtection>
      <deviceType>interactive</deviceType>
   </device>
   <domain>
      <domainID>dhf5434jddAdfD</domainID>
   </domain>
   <serviceOperatorCentre>
      <socID>12345</socID>
      <socInfoURL>http://www.soc.com/info</socInfoURL>
      <socKeyURL>http://www.soc.com/keys</socKeyURL>
      <riID>4567</riID>
      <riURL>http://www.soc.com/ri</riURL>
   </serviceOperatorCentre>
   <purchaseItemList>
```

```
      <purchaseItem>
         <itemID>5432</itemID>
         <version>123435</version>
         <purchaseOption>B</purchaseOption>
         <price currency="EUR">5.00</price>
      </purchaseItem>
   </purchaseItemList>
   <signature>h8twx3rvEPO0vKtMup4NbeVu0flO</signature>
</purchaseRequest>
```

12.2.4.6 Purchase response

12.2.4.6.1 General

The purchase response contains a global status code ("success" or "error") and an optional global failure code in the case that the transaction failed completely. The response includes a list of item-specific status codes and in the case of a subscription to a service that requires RO renewal, the last day and time of validity of the RO per subscription is given.

12.2.4.6.2 XML schema

```
 <xs:element name="purchaseResponse" type="c18:purchaseResponseType"/>
 <xs:complexType name="purchaseResponseType">
   <xs:complexContent>
     <xs:extension base="c18:responseType">
       <xs:sequence>
         <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
         <xs:element name="purchaseItemList">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="purchaseItem" maxOccurs="unbounded">
                 <xs:complexType>
                   <xs:sequence>
                     <xs:element name="reasonCode" type="xs:nonNegativeInteger"
minOccurs="0"/>
                     <xs:element name="roValidityEndTime" type="xs:dateTime" minOccurs="0"/>
                   </xs:sequence>
                   <xs:attribute name="itemID" type="xs:token" use="required"/>
                 </xs:complexType>
               </xs:element>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
       </xs:sequence>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
```


12.2.4.6.3 Example: successful purchase response with RO acquisition trigger

The successful purchase response is a multi-part message, including a RO acquisition trigger (not shown) for each of the requested purchase items.

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseResponse status="success"/>
```

12.2.4.6.4 Example: unsuccessful purchase response with registration trigger

If the device is not registered, the unsuccessful purchase response is a multi-part message, containing a registration trigger (not shown):

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseResponse status="error" reasonCode="3"/>
```

12.2.4.6.5 Example: unsuccessful purchase response with purchase-item-specific error

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseResponse status="error">
 <serviceOperatorCentre>
   <socID>12345</socID>
 </serviceOperatorCentre>
 <purchaseItemList>
   <purchaseItem itemID="5432">
   <reasonCode>1</reasonCode>
   </purchaseItem>
 </purchaseItemList>
</purchaseResponse>
```

12.2.4.7 Subscription RO renewal request

12.2.4.7.1 General

The subscription renewal request includes the user, device and SOC data specified above, and a list of the identifiers of purchase items corresponding to subscriptions the user wants to renew.

12.2.4.7.2 XML schema

```
 <xs:element name="renewalRequest" type="c18:renewalRequestType"/>
 <xs:complexType name="renewalRequestType">
   <xs:complexContent>
     <xs:extension base="c18:requestType">
       <xs:sequence>
         <xs:element name="user" type="c18:userDataType"/>
         <xs:element name="device" type="c18:deviceDataType"/>
         <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
         <xs:element name="purchaseItemList">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="purchaseItem" type="c18:purchaseItemType"
maxOccurs="unbounded"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
         <xs:element name="signature" type="xs:base64Binary"/>
       </xs:sequence>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
```

12.2.4.7.3 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<renewalRequest interfaceVersion="1">
 <user>
   <userID>24403123456</userID>
   <lang>en</lang>
 </user>
 <device>
   <deviceID>0044005817853</deviceID>
   <riDeviceID>1234567890</riDeviceID>
   <bcastBearer>dvbh</bcastBearer>
   <serviceProtection>ipsec</serviceProtection>
   <deviceType>interactive</deviceType>
 </device>
 <serviceOperatorCentre>
   <socID>12345</socID>
   <socKeyURL>http://www.soc.com/keys</socKeyURL>
    <riID>4567</riID>
    <riURL>http://www.soc.com/ri</riURL>
 </serviceOperatorCentre>
 <purchaseItemList>
   <purchaseItem>
     <itemID>5432</itemID>
```

```
   </purchaseItem>
 </purchaseItemList>
 <signature>q2ewx3rvEPO0vKtMup4NbeVu1wd9</signature>
</renewalRequest>
```

12.2.4.8 Subscription RO renewal response

12.2.4.8.1 General

The subscription renewal response contains a global status code ("success" or "error") and an optional global failure code in case the transaction failed completely. The response includes a list of item-specific status codes the updated last day and time of validity of the RO after the renewal.

12.2.4.8.2 Schema

```
<xs:element name="renewalResponse" type="c18:renewalResponseType"/>
<xs:complexType name="renewalResponseType">
  <xs:complexContent>
    <xs:extension base="c18:responseType">
      <xs:sequence>
        <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
        <xs:element name="purchaseItemList">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="purchaseItem" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="reasonCode" type="xs:nonNegativeInteger" minOccurs="0"/>
                    <xs:element name="roValidityEndTime" type="xs:dateTime" minOccurs="0"/>
                  </xs:sequence>
                  <xs:attribute name="itemID" type="xs:token"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```


12.2.4.8.3 Example: successful renewal response with RO acquisition trigger

The successful renewal response is a multi-part message, including a RO acquisition trigger (not shown) for each of the purchase items for which the RO is renewed.

```
<?xml version="1.0" encoding="UTF-8"?>
<renewalResponse status="success">
 <serviceOperatorCentre>
   <socID>12345</socID>
 </serviceOperatorCentre>
 <purchaseItemList>
   <purchaseItem itemID="5432">
     <roValidityEndTime>2005-02-19T00:59:59Z</roValidityEndTime>
   </purchaseItem>
 </purchaseItemList>
</renewalResponse>
```

12.2.4.8.4 Example: unsuccessful renewal response with registration trigger

If the device is not registered, the unsuccessful renewal response is a multi-part message, containing a registration trigger (not shown).

```
<?xml version="1.0" encoding="UTF-8"?>
<renewalResponse status="error" reasonCode="3"/>
```

12.2.4.8.5 Example: unsuccessful renewal response with purchase-item-specific error

If individual items cannot be found, the unsuccessful renewal response contains purchase-item-specific reason codes.

```
<?xml version="1.0" encoding="UTF-8"?>
<renewalResponse status="error">
 <serviceOperatorCentre>
   <socID>12345</socID>
 </serviceOperatorCentre>
 <purchaseItemList>
   <purchaseItem itemID="5432">
       <reasonCode>7</reasonCode>
   </purchaseItem>
 </purchaseItemList>
</renewalResponse>
```

12.2.4.9 Subscription cancellation request

12.2.4.9.1 General

The subscription cancellation request includes the user, device and SOC data specified above, and a list of the identifiers of purchase items corresponding to subscriptions the user wants to cancel.

12.2.4.9.2 XML schema

```
 <xs:element name="cancellationRequest" type="c18:cancellationRequestType"/>
 <xs:complexType name="cancellationRequestType">
   <xs:complexContent>
     <xs:extension base="c18:requestType">
       <xs:sequence>
         <xs:element name="user" type="c18:userDataType"/>
         <xs:element name="device" type="c18:deviceDataType"/>
         <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
         <xs:element name="purchaseItemList">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="purchaseItem" type="c18:purchaseItemType"
maxOccurs="unbounded"/>
             </xs:sequence>
           </xs:complexType>
         </xs:element>
         <xs:element name="signature" type="xs:base64Binary"/>
       </xs:sequence>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
```

12.2.4.9.3 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<cancellationRequest interfaceVersion="1">
 <user>
   <userID>24403123456</userID>
   <lang>en</lang>
 </user>
 <device>
   <deviceID>0044005817853</deviceID>
   <riDeviceID>1234567890</riDeviceID>
   <bcastBearer>dvbh</bcastBearer>
   <serviceProtection>srtp</serviceProtection>
   <deviceType>interactive</deviceType>
 </device>
 <serviceOperatorCentre>
   <socID>12345</socID>
 </serviceOperatorCentre>
 <purchaseItemList>
   <purchaseItem>
     <itemID>5432</itemID>
   </purchaseItem>
```

```
 </purchaseItemList>
 <signature>w2jwx3rvEPO0vKtMup4NbeVu9en1</signature>
</cancellationRequest>
```

12.2.4.10 Subscription cancellation response

12.2.4.10.1 General

The subscription cancellation response contains a global status code ("success" or "error") and an optional global failure code in case the transaction failed completely. In case of success, the response includes a text per purchase item that tells the user when the cancellation takes effect, i.e. from when on the ROs cannot be renewed any more.

The device is expected to continue renewing ROs until the renewal fails with a "no subscription" error message.

12.2.4.10.2 Schema

```
<xs:element name="cancellationResponse" type="c18:cancellationResponseType"/>
<xs:complexType name="cancellationResponseType">
  <xs:complexContent>
    <xs:extension base="c18:responseType">
      <xs:sequence>
        <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
        <xs:element name="purchaseItemList">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="purchaseItem" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                   <xs:element name="reasonCode" type="xs:nonNegativeInteger" minOccurs="0"/>
                   <xs:element name="cancellationInfoMessage" type="xs:string" minOccurs="0"/>
                  </xs:sequence>
                  <xs:attribute name="itemID" type="xs:token"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```


12.2.4.10.3 Example: successful cancellation response

```
<?xml version="1.0" encoding="UTF-8"?>
<cancellationResponse status="success">
 <serviceOperatorCentre>
   <socID>12345</socID>
 </serviceOperatorCentre>
 <purchaseItemList>
   <purchaseItem itemID="5432">
      <cancellationInfoMessage>Service valid until 2005-03-27</cancellationInfoMessage>
   </purchaseItem>
 </purchaseItemList>
</cancellationResponse>
```

12.2.4.10.4 Example: unsuccessful cancellation response with purchase-item-specific error

```
<?xml version="1.0" encoding="UTF-8"?>
<cancellationResponse status="error">
 <serviceOperatorCentre>
   <socID>12345</socID>
 </serviceOperatorCentre>
 <purchaseItemList>
   <purchaseItem itemID="5432">
       <reasonCode>7</reasonCode>
```

```
   </purchaseItem>
 </purchaseItemList>
</cancellationResponse>
```

12.2.4.11 Token request

12.2.4.11.1 General

In addition to the user, device and SOC data specified above, the token request includes

ChargingType: the type of charging (pre-paid or post-paid) the user wishes to use. The SubMgmt will verify that the requested charging type is available for this user;

RequestedTokens: the amount of new tokens requested by the device. In case of pre-paid, the amount of tokens requested is subtracted from the user's credit. In case of post-paid, it is verified that the amount of tokens requested does not exceed the user's credit limit.

12.2.4.11.2 XML schema

```
 <xs:element name="tokenRequest" type="c18:tokenRequestType"/>
 <xs:complexType name="tokenRequestType">
   <xs:complexContent>
     <xs:extension base="c18:requestType">
       <xs:sequence>
         <xs:element name="user" type="c18:userDataType"/>
         <xs:element name="device" type="c18:deviceDataType"/>
         <xs:element name="serviceOperatorCentre" type="c18:serviceOperatorCentreType"/>
         <xs:element name="paymentType">
           <xs:simpleType>
             <xs:restriction base="xs:token">
               <xs:enumeration value="prePaid"/>
               <xs:enumeration value="postPaid"/>
             </xs:restriction>
           </xs:simpleType>
         </xs:element>
         <xs:element name="requestedTokens" type="xs:nonNegativeInteger"/>
         <xs:element name="signature" type="xs:base64Binary"/>
       </xs:sequence>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
```

12.2.4.11.3 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<tokenRequest interfaceVersion="1">
<user>
   <userID>24403123456</userID>
   <lang>en</lang>
 </user>
 <device>
   <deviceID>C0044005817853</deviceID>
   <riDeviceID>1234567890</riDeviceID>
   <bcastBearer>DVBH</bcastBearer>
   serviceProtection>IPsec</serviceProtection>
   <deviceType>interactive</deviceType>
 </device>
 <serviceOperatorCentre>
   <socID>12345</socID>
 </serviceOperatorCentre>
 <paymentType>postPaid</paymentType>
 <requestedTokens>-20</requestedTokens>
 <signature>d2jwx3rvEPO0vKtMup4NbeVu9ke7</signature>
</tokenRequest>
```

12.2.4.12 Token response

12.2.4.12.1 General

The token response reports the success or failure of the operation and includes a ROAP trigger for token acquisition as additional HTTP payload (not shown here).

12.2.4.12.2 Schema

```
 <xs:element name="tokenResponse" type="c18:tokenResponseType"/>
 <xs:complexType name="tokenResponseType">
   <xs:complexContent>
     <xs:extension base="c18:responseType"/>
   </xs:complexContent>
 </xs:complexType>
```

12.2.4.12.3 Example: successful token response

```
<?xml version="1.0" encoding="UTF-8"?>
<tokenResponse  status="success"/>
```

12.2.4.12.4 Example: unsuccessful token response

```
<?xml version="1.0" encoding="UTF-8"?>
<tokenResponse  status="error" failureCode="1"/>
```


12.2.5 XML schema definition for request and response related XML elements

The schema definition below defines the namespace http://www.18crypt.com/2005/XMLSchema and all XML elements used for requests and response. The schema SHALL take precedence over XML elements declared previously in this clause.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:c18="http://www.18crypt.com/2005/XMLSchema"
targetNamespace="http://www.18crypt.com/2005/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:complexType name="userDataType">
        <xs:sequence>
            <xs:element name="userID" type="xs:string"/>
            <xs:element name="lang" type="xs:language" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="deviceDataType">
        <xs:sequence>
            <xs:element name="deviceID" type="xs:token"/>
            <xs:element name="riDeviceID" type="xs:token"/>
            <xs:element name="bcastBearer" type="xs:token" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="serviceProtection" type="xs:token" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="deviceType" type="xs:token" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="domainType">
        <xs:sequence>
            <xs:element name="domainID" type="xs:token"/>
            <xs:element name="domainType" type="xs:token" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="serviceOperatorCentreType">
        <xs:sequence>
            <xs:element name="socID" type="xs:token"/>
            <xs:element name="socInfoURL" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="socKeyURL" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="riID" type="xs:token" minOccurs="0"/>
            <xs:element name="riURL" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="riProxyURL" type="xs:anyURI" minOccurs="0"/>
```

```
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="priceType">
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attribute name="currency" type="xs:token" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="purchaseItemType">
        <xs:sequence>
            <xs:element name="itemID" type="xs:token"/>
            <xs:element name="version" type="xs:nonNegativeInteger" minOccurs="0"/>
            <xs:element name="purchaseOption" type="xs:token" minOccurs="0"/>
            <xs:element name="price" type="c18:priceType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="requestType">
        <xs:attribute name="interfaceVersion" type="xs:nonNegativeInteger" use="required"/>
    </xs:complexType>
    <xs:complexType name="responseType">
        <xs:sequence>
            <xs:element name="message" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="status" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:enumeration value="success"/>
                    <xs:enumeration value="error"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="reasonCode" type="xs:nonNegativeInteger" use="optional"/>
    </xs:complexType>
    <xs:element name="pricingRequest" type="c18:pricingRequestType"/>
    <xs:complexType name="pricingRequestType">
        <xs:complexContent>
            <xs:extension base="c18:requestType">
                <xs:sequence>
                    <xs:element name="user" type="c18:userDataType"/>
                    <xs:element name="device" type="c18:deviceDataType"/>
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="purchaseItemList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="purchaseItem" type="c18:purchaseItemType"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="signature" type="xs:base64Binary"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="pricingResponse" type="c18:pricingResponseType"/>
    <xs:complexType name="pricingResponseType">
        <xs:complexContent>
            <xs:extension base="c18:responseType">
                <xs:sequence minOccurs="0">
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="purchaseItemList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="purchaseItem" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:choice>
                                            <xs:element name="purchaseOptionList">
                                                <xs:complexType>
                                                    <xs:sequence>
                                                        <xs:element name="purchaseOption"
maxOccurs="unbounded">
                                                            <xs:complexType>
                                                                <xs:sequence>
                                                                    <xs:element
name="optionID" type="xs:token"/>
```

```
                                                                    <xs:element name="desc"
type="xs:string"/>
                                                                    <xs:element name="price"
type="c18:priceType"/>
                                                                </xs:sequence>
                                                                <xs:attribute name="type"
use="required">
                                                                    <xs:simpleType>
                                                                        <xs:restriction
base="xs:token">
                                                                            <xs:enumeration
value="one-time"/>
                                                                            <xs:enumeration
value="continuous"/>
                                                                        </xs:restriction>
                                                                    </xs:simpleType>
                                                                </xs:attribute>
                                                            </xs:complexType>
                                                        </xs:element>
                                                    </xs:sequence>
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:element name="reasonCode"
type="xs:negativeInteger"/>
                                        </xs:choice>
                                        <xs:attribute name="itemID" type="xs:token"/>
                                        <xs:attribute name="version"
type="xs:nonNegativeInteger"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="purchaseRequest" type="c18:purchaseRequestType"/>
    <xs:complexType name="purchaseRequestType">
        <xs:complexContent>
            <xs:extension base="c18:requestType">
                <xs:sequence>
                    <xs:element name="user" type="c18:userDataType"/>
                    <xs:element name="device" type="c18:deviceDataType"/>
                    <xs:element name="domain" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="domainID"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="purchaseItemList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="purchaseItem" type="c18:purchaseItemType"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="signature" type="xs:base64Binary"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="purchaseResponse" type="c18:purchaseResponseType"/>
    <xs:complexType name="purchaseResponseType">
        <xs:complexContent>
            <xs:extension base="c18:responseType">
                <xs:sequence>
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="purchaseItemList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="purchaseItem" maxOccurs="unbounded">
                                    <xs:complexType>
```

```
                                        <xs:sequence>
                                            <xs:element name="reasonCode"
type="xs:nonNegativeInteger" minOccurs="0"/>
                                            <xs:element name="roValidityEndTime"
type="xs:dateTime" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attribute name="itemID" type="xs:token"
use="required"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="renewalRequest" type="c18:renewalRequestType"/>
    <xs:complexType name="renewalRequestType">
        <xs:complexContent>
            <xs:extension base="c18:requestType">
                <xs:sequence>
                    <xs:element name="user" type="c18:userDataType"/>
                    <xs:element name="device" type="c18:deviceDataType"/>
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="purchaseItemList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="purchaseItem" type="c18:purchaseItemType"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="signature" type="xs:base64Binary"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="renewalResponse" type="c18:renewalResponseType"/>
    <xs:complexType name="renewalResponseType">
        <xs:complexContent>
            <xs:extension base="c18:responseType">
                <xs:sequence>
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="purchaseItemList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="purchaseItem" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="reasonCode"
type="xs:nonNegativeInteger" minOccurs="0"/>
                                            <xs:element name="roValidityEndTime"
type="xs:dateTime" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attribute name="itemID" type="xs:token"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="cancellationRequest" type="c18:cancellationRequestType"/>
    <xs:complexType name="cancellationRequestType">
        <xs:complexContent>
            <xs:extension base="c18:requestType">
                <xs:sequence>
                    <xs:element name="user" type="c18:userDataType"/>
                    <xs:element name="device" type="c18:deviceDataType"/>
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="purchaseItemList">
                        <xs:complexType>
```

```
                            <xs:sequence>
                                <xs:element name="purchaseItem" type="c18:purchaseItemType"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="signature" type="xs:base64Binary"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="cancellationResponse" type="c18:cancellationResponseType"/>
    <xs:complexType name="cancellationResponseType">
        <xs:complexContent>
            <xs:extension base="c18:responseType">
                <xs:sequence>
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="purchaseItemList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="purchaseItem" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="reasonCode"
type="xs:nonNegativeInteger" minOccurs="0"/>
                                            <xs:element name="cancellationInfoMessage"
type="xs:string" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attribute name="itemID" type="xs:token"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="tokenRequest" type="c18:tokenRequestType"/>
    <xs:complexType name="tokenRequestType">
        <xs:complexContent>
            <xs:extension base="c18:requestType">
                <xs:sequence>
                    <xs:element name="user" type="c18:userDataType"/>
                    <xs:element name="device" type="c18:deviceDataType"/>
                    <xs:element name="serviceOperatorCentre"
type="c18:serviceOperatorCentreType"/>
                    <xs:element name="paymentType">
                        <xs:simpleType>
                            <xs:restriction base="xs:token">
                                <xs:enumeration value="prePaid"/>
                                <xs:enumeration value="postPaid"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="requestedTokens" type="xs:nonNegativeInteger"/>
                    <xs:element name="signature" type="xs:base64Binary"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="tokenResponse" type="c18:tokenResponseType"/>
    <xs:complexType name="tokenResponseType">
        <xs:complexContent>
            <xs:extension base="c18:responseType"/>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```


12.3 Purchase for mixed-mode devices

The sequences specified in 12.2.2 apply also to mixed-mode devices. However, for mixed-mode devices the RI may decide to deliver a BCRO over a broadcast channel as a result of a successful purchase. In this case, the device will not receive a ROAP trigger, but instead it

shall prepare to receive BCROs, which are specified in 8.4.2. The flows in 12.2.2 are thus modified as follows.

Unsuccessful purchase (Figure 45, step 16): after the ROAP registration, the RI may send registration data via the PDR protocol to the device, in the case the device has not registered before.

Successful purchase (Figure 46, step 12): the device receives the "success" response but no RO acquisition trigger. The device shall prepare to receive a BCRO.

RO renewal (Figure 47, step 9): the device receives the "success" response but no RO acquisition trigger. The device shall prepare to receive a BCRO.

Token request (Figure 49, step 8): the device receives the "success" response but no RO acquisition trigger. The device shall prepare to receive a token BCRO.

A mixed-mode device may join a broadcast domain, and domain ROs for that domain may be delivered via an interactivity channel. In this case, the ICRO will be addressed to the long-form domain ID (identical to OMA domain ID) and the device shall obtain the short-form domain ID from the association between long-form and short-form domain IDs it has established when it received the registration data (see 9.3.3).

12.4 Out-of-band purchase

12.4.1 Means of purchase – Introduction

Out-of-band purchase is needed in order to make the purchasing functionality available to broadcast devices. Interactive devices may also use out-of-band purchase, for the broadcast mode of operation.

Figure 40 shows the dataflow that governs the broadcast mode of operation. Device registration as well as the transactions related to purchasing require that information is conveyed from the device or its user to the network.

There are numerous ways how out-of-band communication can be implemented (for example, web shop, call centre, automated SMS service, etc.). A viable operation would be where the user of the device calls the COC and tells that he wants to see the football match of tonight. Such options are not mandated.

The following subclause specifies the behaviour of the device in case it is equipped with an appropriate contact() either via the service guide or via the update_contact_number_msg() message.

12.4.2 Out-of-band purchase from service guide data

Generally, the request/response pairs for purchase over an interactivity channel need to be performed out-of-band. If the out-of-band communication requires that the user speaks or types some information, it is desirable that instead of long numeric identifiers, shorter (properly scoped) numbers, codes, or names can be used. These "human-friendly" numbers and names are expected to be part of the service guide, or, if they are used to identify the device, be put in place during device registration. In case the short version of an identifier is not available in the service guide, the long version will be used instead. The service guide will also include a textual representation of the customer operation centre contact information and the out-of-band channels available, to be displayed to the user. These numbers, codes and names are established in 12.5.

In order to request a purchase, the user is required to provide the data in Table 108 to the customer operation centre, via the out-of-band channel indicated by the service guide.

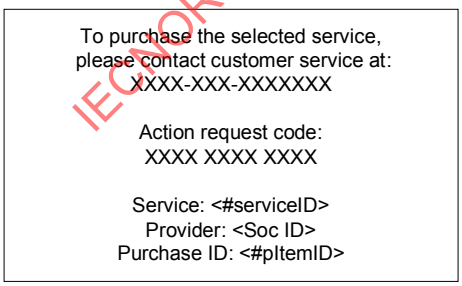
Table 108 – Data to be provided to the customer operation centre

Data	Short concise version	Human friendly version
contact() "number"	Local phone number or international phone number or SMS number or URL	Local phone number or international phone number or SMS number or URL
Device ID	ARC code (see A.9.3) including purchase code	ARC code (see A.9.3) including purchase code
Service ID	Service ID code (i.e. #serviceID)	Service name (i.e. serviceName)
Service Operator Centre ID	Soc ID (i.e. socID)	Soc name (i.e. socName)
Purchase Item ID	Purchase item code (i.e. #pItemName)	Purchase item name (i.e. PItemName)
NOTE 1 The data in Table 108 is provided by the service guide information; see 12.5 for details. NOTE 2 A device that has not been registered before will show the UDN instead of the ARC.		

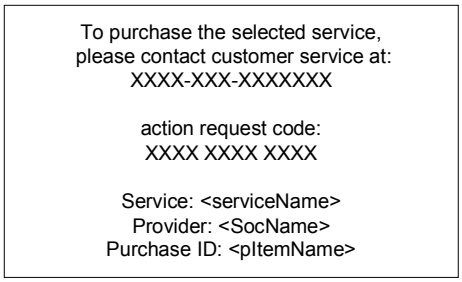
During the out-of-band purchase, the device may display a dialogue with instructions. Notifying the device data can be done in various ways, for example by showing the user of the device a dialogue on the screen of the device, displaying the data necessary for purchase and a contact() "number" that needs to be contacted. A phone number may be used for vocal notification of the data to the RI. Another example is to display instructions to send an SMS message via a mobile phone to the RI. Yet another example is to display a URL, which may be used by the user to contact a web shop via an offline browser.

An example of a displayed OOB purchase message for a registered device is shown in Figure 50, where the information to be reported back to the RI is shown.

It is the sequence of the defined values that is specified. The use of dashes as the delimiter is shown with an example placement to be consistent with the examples used elsewhere in this standard. The text portion of this screen is shown as an example only; there is no implied requirement to duplicate the exact wording or formatting shown. The numeric fields shall be included as defined above. The ARC code will only be displayed after the first registration, when that data may available for display.



An example dialogue concise instructions for vocal notification of ARC and purchase data to the call centre.



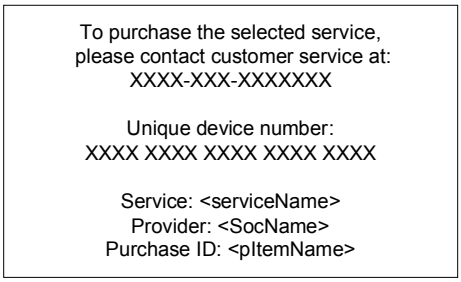
An example dialogue showing human friendly instructions for vocal notification of ARC and purchase data to the call centre.

Key

- The contact can be any number as specified in contact() (see 9.3.2.7.6.3) or delivered by the service guide.
- The action request code shall incorporate the purchase action request code (see 9.3.2.3).

Figure 50 – Samples of out-of-band purchase information displays for a registered device

The dialogue shown in Figure 51 is an example of what an unregistered device could display.



An example dialogue showing human friendly instructions for vocal notification of UDN and purchase data to the call centre.

Figure 51 – Sample of out-of-band purchase information displays for an unregistered device

NOTE 2 An unregistered device performing a purchase request will need to register first. See 9.3.2 for details.

The result of a successful purchase is usually a BCRO. For its acquisition, the standard 2-pass ROAP cannot be used. Instead an RI service is defined (see Clause 10), that comprises mechanisms to broadcast BCROs spontaneously (as a result of the purchase transaction) or in a scheduled manner (for example, for re-keying the SEAKs).

12.5 Required service guide information

12.5.1 General

This standard fully specifies extensions to the service and content guides of the following systems to meet the signalling and service guide requirements:

- IPDC over DVB-H (in 13.6.3);
- DVB-T/C/S (in 14.6);
- DVB-IP (in 15.7 and 16.6).

It is also possible to use this standard with other service guides. In this case, this standard does not prescribe the precise structure of the service guide, but the purchasing part of the service guide needs to be designed in line with the requirements derived from this subclause, which deals with purchasing (the attributes as presented above enable purchasing only if the relationships between the entities in the service guide are in line with the architectural model).

The required attributes are specified in 12.5.2 to 12.5.8.

12.5.2 **Service operation centre** (including service distribution management)

socID (mandatory, 1) – Is the globally co-ordinated ID of the service operation centre (a.k.a. "socID"). In case of IPDC systems, the DVB platform ID may (and is recommended to) be used for this purpose. For DVB-T/C/S systems, it is recommended to use the original network id. The ID has to comply with the definition in http://www.w3.org/TR/xmlschema-2/#ID.

socName (mandatory, 1) – Is the name of the service operation centre, reasonably globally unique (for example, this can be achieved by concatenating of the ISO country code and an acronym that identifies the operator within the country). This name may be used for identifying the SOC operator within out-of-band purchase transactions. In XML documents, the name shall be encoded using UTF-8, see IETF RFC 3629.

socKeyURL (mandatory, 1) – URL through which an RI can retrieve keys from the service distribution management.

socInfoURL (mandatory, 1) – URL through which the service subscription management can retrieve purchase info from the service distribution management.

socRiProxyURL (optional, 1) – URL of the RI Proxy associated with the service operation centre (used by a service subscription management in a different network to insert BCROs, as specified in B.8.3). This parameter is present only if the RI Proxy functionality is supported by the SOC.

12.5.3 **Customer operation centre** (including service subscription management)

cocID (mandatory, 1) – Is the globally coordinated ID of the customer operation centre.

cocLocalFlag (mandatory, 1) – indicates, if "true", that a COC is local to the SOC and therefore advertises the availability and purchase information completely in the service guide. This knowledge helps avoiding unnecessary requests by the device to obtain information whether or not a particular purchase item is available from a certain COC (not all items will, in general, be purchasable through all COCs).

cocName (mandatory, 1, multi-language) – Is the name of the customer operation centre, reasonably globally unique (for example, this can be achieved by concatenating of the ISO country code and an acronym that identifies the operator within the country). This name may be used for identifying the COC operator within out-of-band purchase transactions. The name shall be encoded using UTF-8; see IETF RFC 3629.

cocRekeyingSafetyWindow (mandatory, 1) – Is the time interval DT, as specified in 12.2.2.8, during which it cannot be guaranteed that RO renewal will succeed timely for uninterrupted access to a particular service.

cocPurchaseURL (optional, 0..1) – Specifies the URL for a pricing or purchase or subscription RO renewal request can be sent by the device over HTTP POST or HTTPS POST. If present, all the purchase transactions that are specified in this standard shall be supported.

cocPortalURL (optional, 0..1) – Specifies the URL on which customer operation centre may offer out-of-band self-provisioning via HTTP(S). If present, the portal shall support all the out-of-band purchase transactions that are specified in this standard.

cocContactInfo (optional, 0..n, multi-language) – Is a text string that indicates to a user how to contact a COC to initiate an out-of-band purchase transaction (for example, toll-free phone number, international phone number, letter address, e-mail address, SMS number, etc.).

cocRiID (mandatory, 1) – ID of the RI associated with the customer operation centre (needed to allow broadcast devices to identity the RI service that may be operated by their home COC).

cocRiURL (mandatory, 1) – URL of the RI associated with the customer operation centre (used by service subscription management for requesting ROAP triggers or requesting registration data or BCROs to be broadcast).

The service guide shall include an entry for every COC with which the SOC has an agreement ("locality agreement", "roaming agreement"; see Clause B.8). For "local COCs", the service guide shall include the complete availability and purchase information, and should include also the pricing information (purchase options with their respective prices).

12.5.4 Service

serviceID (mandatory, 1) – Is the ID of the service, unique within the scope of a service distribution management (socID).

serviceTypeEnum (mandatory, 1) – Is the type of the service. The allowed values are:

"media" – the service is a regular media service;

"sg" – the service is a service guide, and its streams carry service guide objects;

"ri" – the service is an RI service, and its streams carry RI objects (registration data, BCROs, RI triggers, RI messages) for the broadcast mode of operation.

serviceName (mandatory, 1, multi-language) – Is the name of the service, unique within the scope of a service distribution management (socID). The service name may be used for identifying the service within out-of-band purchase transactions. In XML documents, the name shall be encoded using UTF-8, see IETF RFC 3629.

serviceDescription (optional, 0..1, multi-language) – Is the description of the service.

serviceBaseCID (mandatory, 1) – Is used as part of the CID of all service and programme keys related to the service.

NOTE ServiceBaseCID is not required for non-encrypted services.

12.5.5 ScheduleItem

sItemID (mandatory, 1) – Is the ID of the schedule item (programme), unique within the scope of a service distribution management (socID).

sItemName (mandatory, 1, multi-language) – Is the user-friendly name that is used to identify the schedule item, unique within the scope of a service distribution management (socID). The schedule item name may be used for identifying the schedule item within out-of-band purchase transactions. In XML documents, the name shall be encoded using UTF-8, see IETF RFC 3629.

sItemDescription (optional, 0..1, multi-language) – Is the description of the schedule item.

sItemStartTime – Is the start time of the schedule item. In XML documents, it is specified as yyyy-mm-ddThh:mm:ss

sItemEndTime – Is the end time of the schedule item. In XML documents, it is specified as yyyy-mm-ddThh:mm:ss

12.5.6 ContentItem

cItemID (mandatory, 1) – Is the ID of the content item, unique within the scope of a service distribution management (socID).

cItemTypeEnum (mandatory, 1) – Is the type of the content item. The allowed values are:

"media" – the content item is a regular media item;

"riSet" – the content item is defining a set of devices using the broadcast mode of operation.

cItemName (mandatory, 1, multi-language) – Is the user-friendly name that is used to identify the content item, unique within the scope of a service distribution management (socID). The schedule item name may be used for identifying the content within out-of-band purchase transactions. In XML documents, the name shall be encoded using UTF-8; see IETF RFC 3629.

cItemDescription (optional, 0..1, multi-language) – Is the description of the content item. If, and only if, the type of the content item is "riSet", then the description is a structured mono-language attribute with the following substructure:

cItemGroupRange (optional, 0..n) – Is a range of subscriber groups, specified as a tuple (low, high);

cItemDeviceRange (optional, 0..n) – Is a range of individual device IDs, specified as a tuple (low, high);

cItemDomainRange (optional, 0..n) – Is a range of domain IDs, specified as a tuple (low, high).

12.5.7 Purchase item

pItemID (mandatory, 1) – Is the ID of the purchase item (a.k.a. "purchase_item_id"), unique within the scope of a service distribution management (socID).

pItemVersion (mandatory, 1) – Is the version number of the purchase item, used to verify that the purchase item data received via the service guide is synchronized with the data in the subscription management.

pItemName (mandatory, 1, multi-language) – Is the user-friendly name that is used to identify the service, unique within the scope of a service distribution management (socID). The purchase item name may be used for identifying the purchase item within out-of-band purchase transactions. In XML documents, the name shall be encoded using UTF-8, see IETF RFC 3629.

pItemDescription (optional, 0..1, multi-language) – Is the description of the service, used for the purposes of the user.

pItemServiceID (optional, 0..n) – Is the serviceID of a service that is part of the purchase item (which is in this case a subscription item to a service bundle and is not intended to have any schedule or content items associated).

pItemScheduleItemID (optional, 0..1) – Is the ID of a schedule item (sItemID) that is part of the purchase item (which is in this case a pay-per-view purchase for a programme and is not intended to have any services or content items associated).

pItemContentItemID (optional, 0..n) – Is the ID of a content item (cItemID) that is part of the purchase item (which should in this case not have any services or schedule items associated).

12.5.8 Purchase data

cocID (mandatory, 1) – Is the ID of the customer operation centre.

pItemID (mandatory, 1) – Is the ID of the purchase item.

purchaseOption (optional, 0..n) – Is a structure including all the information (purchase option code [unique within purchase data record], description [multi-language], price [with currency], type [continuous or one-time], availability for domain subscription and for which domain type; see OMA DRM 2.0, OMA XBS or both) that is needed for the user to decide upon a purchase, and for the device to execute a purchase (i.e. to make a purchase request); this should essentially be the same information as the information returned to the device in a successful pricing response.

For a "local COC" the availability information shall be complete, i.e. for all purchase items that are available through the COC, there shall be a purchase data fragment in the service guide, which shall include also all purchase options and their respective pricing.

13 Protection of IPDC over DVB-H systems

13.1 General

This clause specifies the use of specific protection technologies from Clauses 6, 7, 8, 9, 10 and 11 for IP data casting over DVB-H (see ETSI EN 302 304 and Clause B.11 for general references to these systems). Clauses 4, 5 and 12 are general and apply to all systems.

13.2 Delivery of traffic layer data in IPDC over DVB-H systems

Table 109 shows how the traffic layer options shall be used when traffic layer data is transmitted over IPDC over DVB-H.

Table 109 – Traffic layer options for transmission over IPDC over DVB-H

Traffic layer	Transmission details
IPsec	As specified in IETF RFC 4301
ISMACryp	Not applicable
SRTP	As specified in IETF RFC 3711 and IETF RFC 4771
MPEG2 TS	Not applicable

13.3 Delivery of key stream data in IPDC over DVB-H systems

Each KSM shall be encapsulated in exactly one UDP packet. The KSM data shall be carried in a distinct IP flow, separated from other data.

In order to keep access times low for devices that start accessing a service, a KSM shall be transmitted periodically.

The KSM shall be transported in-band, in the same elementary stream together with the media streams that are protected with the traffic keys contained in the KSM.

13.4 Delivery of rights management data in IPDC over DVB-H systems

13.4.1 General

IPDC over DVB-H systems may use a broadcast channel, an interactivity channel or both a broadcast channel as well as an interactivity channel for the delivery of rights management data (rights objects). The use of broadcast channels and interactivity channels is specified in the next two subclauses.

13.4.2 Delivery of ICROs in IPDC over DVB-H systems over interactivity channel

When an interactivity channel is used to deliver rights objects, all aspects of rights object delivery are governed by OMA DRM 2.0 specifications; see OMA-ERP-DRM-V2_0. Rights objects delivered over an interactivity channel are referred to as interactivity channel rights objects (ICRO).

The interactivity channel used may be a cellular radio network.

13.4.3 Delivery of BCROs in IPDC over DVB-H systems over broadcast channel

BCROs (see 8.4.2) shall be broadcast within an RI service, as defined in Clause 11. The RI service(s) used by individual rights issuers are identified by the service guide.

13.5 Delivery of registration data in IPDC over DVB-H systems

13.5.1 General

IPDC over DVB-H systems may use a broadcast channel or an interactivity channel, or both a broadcast channel as well as an interactivity channel for the delivery of registration data. The use of broadcast channels and interactivity channels is specified in the next two subclauses.

13.5.2 Delivery of registration data in IPDC over DVB-H systems over an interactivity channel

When an interactivity channel is used to deliver registration data, all aspects of delivery are governed by OMA DRM 2.0 specifications; see OMA-ERP-DRM-V2_0. However, see also 9.3.5 for the delivery of broadcast registration data using ROAP.

The interactivity channel used may be a cellular radio network.

13.5.3 Delivery of registration data in IPDC over DVB-H systems over a broadcast channel

When a broadcast channel is used to deliver registration data, registration data shall be broadcast within an RI service, as defined in Clause 11. The RI service(s) used by individual rights issues are identified by the service guide.

13.6 Signalling and service guides in IPDC over DVB-H systems

13.6.1 General

Signalling is provided at two layers: the electronic service guide (ESG) and the session description protocol (SDP), see IETF RFC 2327. The signalling is defined in ETSI TS 102 471 (ESG) and ETSI TS 102 472 (CDP) respectively. The two subclauses below identify the parts of these specifications that are relevant to SPP signalling.

13.6.2 Signalling of KSM in IPDC over DVB-H systems

ETSI TS 102 472 (CDP) specifies how SDP is included in an SDP file.

- ETSI TS 102 472, 10.1, specifies the fields used within an SDP file to signal the presence of a key stream.
- ETSI TS 102 472, 10.2, specifies the fields used within an SDP file to describe a KMM stream.
- ETSI TS 102 472, 10.3, specifies how to bind a key stream to one or more components forming a service.

Additionally, the following attributes shall be included in SDP when SRTP is used:

a = SRTPAuthentication:*n*

where *n* is the SRTP authentication algorithm value for the version of the RCC transform used, as specified in IETF RFC 4771.

a = SRTPROCTxRate:*R*

where *R* is the value of the ROC transmission rate parameter, an integer between 1 and 65535 inclusive, as specified in IETF RFC 4771.

Additionally, if there are multiple KSM streams associated with a single service, which are not distinguishable by IPDCoperatorID, the following attributes shall be provided:

a = srvCIDExt:<x>

where <x> is the value of the most significant 8 bits of service_CID_extension (0…255), and

a = prgCIDExt:<y>

where <y> is the value of the most significant 8 bits of programme_CID_extension (0…255), if used.

Details of the extensions to ETSI TS 102 471 (ESG) required to carry information specific to this standard can be found in Clause A.17.

13.6.3 The service guide for IPDC over DVB-H systems

The signalling provided within the ESG specification ETSI TS 102 471 (ESG) enables the following functionality.

- The inclusion of SPP-specific data for the purchasing of rights to consume content/services.
- The ability to signal the SPP system or systems that will enable consumption of a service or content item.
- The ability to signal a service on which KMS-specific data is delivered.
- The ability to describe details about how and from where rights to the content/services can be obtained.

ETSI TS 102 471 (ESG), 5.4 specifies the service fragment, which may be used to signal to the terminal the presence of a service carrying KMS-specific data. A service carrying KMS-specific data is declared by setting the ServiceType element of the fragment appropriately. Any KMS-specific fields may be included using the PrivateData element of the service fragment.

ETSI TS 102 471 (ESG), 5.8 defines a PurchaseRequestType within the purchase fragment that enables the inclusion of SPP-specific data required to make a purchase. This type has three elements, as follows.

- DRMSystem – This uniquely identifies the KMS from which the purchase can be made.
- PurchaseData – This uses an abstract data type, allowing the inclusion of KMS-specific fields to support the purchasing of the described offer.
- PurchaseChannelIDRef – This is an optional element that enables the referencing of a PurchaseChannel fragment. The PurchaseChannel fragment describes an entity from which the purchase may be made.

A purchase fragment allows multiple PurchaseRequests to be defined, therefore enabling the purchase of content from multiple KMSs.

ETSI TS 102 471 (ESG), 5.9 specifies the PurchaseChannel fragment, and provides the ability to describe a point of purchase. Data specific to a KMS may be included using the PrivateData element.

ETSI TS 102 471 (ESG), 5.10 specifies the acquisition fragment. This fragment defines a KeyStream element that enables the signalling of the KMS and the operator used for the delivery of the content associated with the acquisition fragment.

Details of the extensions to ETSI TS 102 471 (ESG) required to carry information specific to this specification can be found in Clause A.17.

13.7 Format and use of RI streams over IPDC over DVB-H systems

13.7.1 General

All the objects defined in this specification are carried in rights issuer streams. The format of these streams is defined in this subclause.

These streams have the following characteristics.

- The bandwidth overhead of the stream format is minimized.
- Objects of varying sizes (smaller than, similar to, and larger than, the size of an IP packet) are efficiently carried.
- Devices are able to start interpreting the stream at any packet.
- Where packet reception is unreliable or where packets have been reordered, devices are able to determine which objects have been correctly and completely received.

NOTE It is assumed that the underlying IP stack and the layers below it will provide all the necessary error detection and that IP packets received by the service protection system can be assumed to be as transmitted.

13.7.2 IP characteristics

The data carried by IP Datacast systems is logically divided into services. Each rights issuer service consists of one or more IP streams. The following requirements are made.

- A rights issuer stream shall be a distinct IP stream within a service.
- Rights issuer services shall carry only rights issuer streams. It is also allowed for other types of service, including media services, to carry RI streams.
- A rights issuer service may contain any number of rights issuer streams.
- RI services shall be identified as services in the ESG. RI services shall contain only RI streams.
- All RI streams forming part of any service shall be identified as such in the ESG.

Rights issuer streams are IP streams advertised in the ESG. The ESG shall also carry an identifier for the version of this specification used to generate each RI stream. The format of the IP packets is UDP; see IETF RFC 768. This standard does not specify any limits to the length of these IP packets – this will instead be determined by the underlying network.

Subclause 13.7.3 defines the format of the packets of the RI stream.

13.7.3 RI stream packet format

The rights issuer stream is made up of RI stream packets. There shall be at most one RI stream packet per UDP packet, and each UDP packet shall contain only an RI stream packet. The length of the RI stream packet is determined by the broadcaster.

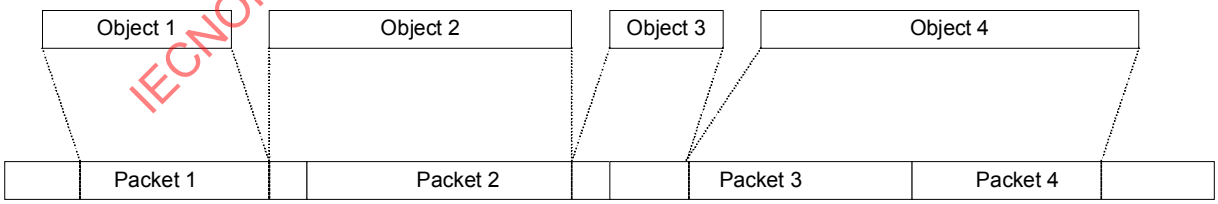


Figure 52 – Example mapping of objects to RI stream packets

Objects are placed into RI stream packets according to the following rules (see Figure 52 for an example).

- If the length of an object and its RI stream header is less than, or equal to, the remaining empty length of a packet, the object is placed in the packet in its entirety and the split_flag is set to zero.
- If the length of an object is greater than the remaining empty length of a packet, the following applies.

  - The object is allocated an object_id.
  - The number of packets required to carry the object is calculated, including the remaining space in the current packet. The part of an object to be placed in each packet is hereafter referred to as a fragment.
  - The object is split into the appropriate fragments. Fragments may be of varying length, for example, if the first fragment of the object begins part way through a packet.
  - While fragments remain to be carried, a header for each fragment is generated, containing the object ID, the number of this fragment within the object and the total number of fragments in the object. The split_flag is set to one.
- Packets shall not contain any empty space. The end of the last bytes within a packet carrying information shall be the end of the packet and the length field of the UDP and IP packet headers will be filled in appropriately. No padding bytes are allowed as part of this protocol.
- The process is repeated with the next object. The size of each packet can be decided by the rights issuer, up to the maximum MTU supported by the network.

The format of the packet is specified in Table 110.

Table 110 – Format of the rights issuer stream

Field	Length	Format
while (bytes left in packet){		
split_flag	1	bslbf
if(split_flag == 1) {		
object_id	7	bslbf
fragment_number_within_object	4	bslbf
total_number_of_fragments_for_object	4	bslbf
if(fragment_number_within_object == total_number_of_fragments_for_object) {		
reserved_for_future_use	4	bslbf
remaining_length_in_packet	12	bslbf
}		
bytes_of_object()		
}		
else{		
reserved_for_future_use	3	bslbf
length_of_object	12	bslbf
bytes_of_object()		
}		
}		

split_flag – If 1, this object is split over multiple packets. If 0, this object is completely contained in this packet.

object_id – An identifier for this object. All fragments of an object (that are carried in separate packets) have the same object ID. This is only required for objects that are split over multiple packets. For each split object generated, this object ID shall be incremented by $1 \bmod (2^7)$.

fragment_number_within_object – The number of this fragment within the object.

total_number_of_fragments_for_object – The total number of fragments that make up the object.

reserved_for_future_use – These bits are reserved for future use and shall be set to 0 in systems according to this standard.

remaining_length_in_packet – The length of the remaining bytes of the current object in this packet.

length_of_object – The length of this object (which is completely contained in this packet).

bytes_of_object() – The bytes of the object to be carried in this packet.

13.7.4 Implementation notes

13.7.4.1 Unreliable delivery

IP networks do not usually offer reliable delivery of packets – this is particularly true of broadcast systems. Devices might not receive all the packets of the RI stream. Where missing packets cause the device to receive only part of an object, the device shall discard this object, although apparently missing packets could later be received due to packet reordering; see 13.7.4.2.

13.7.4.2 Changes in packet order

IP packet order can change between the source and destination hosts on some types of IP network. Of course, this cannot happen on a broadcast link, but it could happen within head-end systems.

At reception time, it is not possible for a device to tell whether an apparently missing packet has been missed due to a reception problem, or whether it will be received later due to some upstream packet reordering. Consider the situation where three packets 1, 2, 3 are reordered and a device receives them in the order 1, 3, 2. When the packet-processing module receives packet number 3, it will appear as if packet 2 has been missed. However, if the device stores packet 3, and then receives and processes packet 2, it can reconstruct all the objects completely contained in all three packets. In order to implement this reconstruction scheme, the device buffers partly received objects for some time and then reconstructs the whole object if the remainder is later received. Incomplete objects are discarded after some period of time. The limit on the use of this technique and the extent of reordering it can cope with is the amount of buffering provided within a device for partly received objects.

The protocol used for rights issuer streams supports the recovery of data from reordered packets as described above. Using the object_id field, it is possible to identify which object a fragment of data belongs to, allowing partly received objects to be buffered and fragments to be allocated to the correct object.

The implementation of any scheme of this kind is not required by this specification. However, it is RECOMMENDED that devices are implemented in such a way that limited packet reordering does not cause objects to be unnecessarily discarded.

It is also RECOMMENDED that service operations centres and rights issuers minimize changes in packet order within their systems.

13.7.4.3 Addressing of objects

The RI stream packet format does not contain addressing information for objects. The format of each object includes addressing information relevant to that object. Devices can determine when an object is addressed to the local device or a group of which the device is a member in the following way.

- The device examines the message tag and the version number of the message to determine what type of message is being broadcast.
- The format of the message contains fields addressing the message to devices in some way. These fields are used to determine whether the local device is being addressed.

13.7.5 Mapping of messages to RI services and streams

13.7.5.1 General

Within an IP datacast network, devices discover streams using the ESG and various SI/PSI tables.

- The ESG maps services to IP addresses, allowing a device to discover what services are available, on which IP stream or streams these services are carried and on which IP addresses these streams can be found.
- SI/PSI data describes how a device can receive IP datacast broadcasts to particular IP addresses, including such information as the PID of the stream carrying the data.

Information about RI services is carried in the same way as for any other service. RI services may contain any number of IP streams. When receiving a service, a device will receive all the streams that make up that service.

IP datacast systems typically use a number of multicast streams to transmit data to receiving devices. It is not anticipated that devices will be allocated individual IP addresses that will then be used to address streams to single devices.

The following subclauses specify how messages are mapped to services and streams.

13.7.5.2 Rights issuer services with complete schedule information

As mentioned above, rights issuers may provide a schedule for the broadcast of messages to sets of devices. If a rights issuer broadcasts a complete schedule of messages to be sent to all devices (excluding *ad hoc* streams), that rights issuer may have any number of RI services, containing any number of RI streams.

For each service, the rights issuer shall broadcast, within the ESG, one or more schedule items containing a list of devices for which messages may be broadcast on that service, and the times at which those messages will be broadcast. Any device registered with the rights issuer shall be able to locate a single RI service to listen to at any one time.

13.7.5.3 Rights issuer services without complete schedule information

If a right issuer broadcasts either no schedule information or incomplete schedule information, that rights issuer shall broadcast only one rights issuer service.

Devices for which schedule information is broadcast should listen at the appropriate times. Devices for which schedule information is not broadcast should listen as often as practical, but no requirements are placed on their behaviour.

13.7.6 Discovery of RI services, streams and schedule Information

The ESG carries information about RI services, streams and their associated schedule information.

The ESG requirements are as follows.

- The capability to describe a service as an RI service and have an associated rights issuer ID.

- The capability to describe a stream within any service as an RI stream and, when the stream is not part of an RI service, have an associated rights issuer ID.
- The capability to describe the standard used to construct an RI stream.
- The capability to include multiple RI schedule blocks, which may be placed within ESG context or also associated with a purchase channel.
- The capability to indicate that a certificate chain update will be included in a particular schedule slot.

For scheduled RI services, a number of particular "content items" are announced, corresponding to device, group or domain ranges (or no range, corresponding to all devices). One of the existing attributes of the content item is for this purpose defined to have a particular structure and semantics.

The broadcast times of these particular content items will be announced within schedule items, and thereby a device will be able to determine when it has to receive the RI service.

13.7.7 Certificate chain updates

It is important that devices can acquire certificate chain updates, which may include an OCSP response, as quickly as possible. A device will not be able to decode services until it has a current certificate chain (although a grace mechanism is defined in A.1.2 to make this more user-friendly). The following requirements are made on the broadcast of certificate chain updates.

- It is strongly recommended that schedule information for certificate chain updates is made available in the ESG. When such schedule information is carried, devices should listen to the relevant RI services when they need to acquire updates. No firm requirement on device behaviour can be made, as a device may not be able to receive a service at a particular time, for example because it has a low battery or is out of range of the broadcast network.

  Signalling of certificate chain updates is specified in Clause A.17.
- Furthermore, it is strongly recommended that a reference to at least the next certificate chain update is always carried in the ESG.
- Where such schedule information is not carried, certificate chain updates shall be carried, at least, in the RI service belonging to the relevant rights issuer.

Using the mechanisms specified in this subclause, two possible schemes for the broadcast of certificate chain updates are informatively described below.

- Certificate chain updates can be broadcast continuously in an RI stream. A schedule block indicating a certificate chain update, with no device range limit and a time limit of, say, midnight to midnight, is broadcast for this stream, indicating that certificate chain updates can always be found on this stream.
- Certificate chain updates are broadcast periodically on an RI stream. Schedule blocks indicating a certificate chain update, with no device range limit and the time limit for when the updates will be broadcast, is broadcast for this stream.

13.7.8 Resending of BCROs

13.7.8.1 General

There is no guarantee that a device will receive the BCROs sent to it via a broadcast channel. A device may request that the BCROs be sent once again by the rights issuer.

13.7.8.2 Resending of BCROs to interactive devices

For an interactive device, requests to resend BCROs can be made via an interactivity channel. If the BCROs are to be delivered via a broadcast channel, the device will listen to the relevant

rights issuer service after sending the request. It is recommended that devices listen to this channel for at least 1 h or until the BCROs are received. It is expected that the BCROs will be delivered in an *ad hoc* RI stream.

When a rights issuer receives a request from an interactive device to resend BCROs over a broadcast channel, it may resend the BCROs for that device.

13.7.8.3 Resending of BCROs to broadcast devices

Rights issuers may allow users of broadcast devices to request that BCROs for that device are resent. If the rights issuer does allow this, the device may prompt the user to make such a request, as specified in 9.3.2.3. The device should then listen to the relevant rights issuer service, possibly after the user has acknowledged that the request has been made. It is recommended that devices listen to this channel for at least 1 h, or until the BCROs are received. It is expected that the BCROs will be delivered in an *ad hoc* RI stream.

No firm requirement on device behaviour can be made, as a device may not be able to receive a service at a particular time, for example, because it has a low battery or is out of range of the broadcast network.

When the rights issuer receives such a request, it may resend the BCROs for that user.

13.7.9 Summary of requirements for rights issuers

If a rights issuer delivers messages to devices via a broadcast channel, it shall use rights issuer services and streams to do so and shall meet the requirements below. If a rights issuer does not deliver messages via a broadcast channel, it will not have rights issuer services and streams, and the remainder of this subclause does not apply.

- Each rights issuer shall either
  - provide a complete schedule for their RI services, covering all registered devices and allowed any registered device to identify one RI service to listen to; or
  - have exactly one rights issuer service.
- Each rights issuer service
  - shall not contain more than three RI streams; and
  - shall contain only rights issuer streams.
- Rights issuers should provide an informative schedule for the broadcast of messages in their RI service, unless the system is being used in an environment where power consumption of devices is not an issue (as the scheduling of RI services is primarily intended as a power-saving feature for devices).
- Any other type of service may carry, at most, one In-band rights issuer stream per rights issuer.
- Rights issuers should broadcast both the current and next rights objects required to receive services, to reduce the likelihood of a device not having the rights object required to receive a service that it is entitled to receive.

13.7.10 Summary of requirements for devices

The following is a summary of the requirements relating to RI services for devices, which support the broadcast mode of operation. None of these requirements apply to devices which only use an interactivity channel to communicate with rights issuers.

- For each rights issuer with which the device is registered, a device shall listen to the associated rights issuer service, subject to the following.
  - Where a schedule for the RI service is available, devices may receive that schedule and may listen to the RI service only at the relevant times.

  - Devices that make use of the schedule should check for new schedule data at least once per day.
- Otherwise, when a schedule for the RI service is not available or a device does not listen to it,
  - mains-powered devices or devices under charge should listen to that service continuously;
  - battery-powered devices should listen to that service at least when the device is powered on for some purpose.

- When receiving a rights issuer service, devices shall listen to all streams within that service.
- It shall be possible to put a device into a mode in which it receives the RI service of a particular rights issuer, for some period, in order to receive, for example, registration data, domain messages and recently purchased rights objects. These are expected to be delivered in *ad hoc* RI streams. This does not apply in the case that a device continuously receives rights issuer services.
- When a device is receiving a service containing an in-band RI stream for a rights issuer with which it is registered, the device shall listen to that stream.

13.7.11 Mapping of messages to DVB-H time sliced bursts

13.7.11.1 General

This subclause specifies how the various key streams and other messages shall be mapped to DVB-H time slicing bursts. For the definition of such a burst, see ETSI EN 301 192.

13.7.11.2 Key stream messages

The following requirements apply to key stream messages.

- Key stream messages shall be carried in the same burst as the content to which the "current" traffic key they carry applies. Key stream messages may carry a second traffic key applying to the next crypto period, which may apply to content in the next burst.
- A key stream message containing the DRM time field should be carried in every burst.
- Within a burst, key stream messages should be broadcast before the content packets, which they are used to decrypt.

13.7.11.3 RI service channel

There are no requirements for the mapping of RI services to DVB-H bursts.

13.7.11.4 In-band RI stream

In-band RI streams are carried as a separate IP stream within a service. As such, they are broadcast within the bursts of that service. Therefore, an in-band RI stream will be received by a device without the need for that device to listen to additional time sliced bursts. This will reduce power consumption by the device compared to delivering the messages via a separate rights issuer service.

14 Protection of DVB T/C/S systems

14.1 General

This clause specifies the use of specific protection technologies from Clauses 6, 7, 8, 9, 10, 11 and 12 for DVB terrestrial, cable and satellite systems (see Clause B.11 for general references to these systems). Clauses 4, 5 and 12 are general and apply to all systems.

In Clause 14, several types of reserved fields are used. In Clause 14, the use of all reserved fields shall be as defined in ETSI EN 300 468, including, but not limited to, the setting of the value of the reserved fields.

NOTE The use of reserved fields in all other clauses of this standard is different.

14.2 Delivery of traffic layer data in DVB T/C/S systems

Table 111 shows how the traffic layer options shall be used when traffic layer data is transmitted over MPEG2 TS-based networks.

Table 111 – Traffic layer options for transmission over MPEG2 TS-based networks

Traffic layer	Transmission details
IPsec	Not applicable
ISMACryp	Not applicable
SRTP	Not applicable
MPEG2 TS	As specified in 6.5

14.3 Delivery of key stream data in DVB T/C/S systems

The table from ETSI ETR 289 shall be used to carry the KSM from 7.2 as payload, as specified in Table 112. The network can identify if the KSM payload carries the odd or even key by reading the conditional "odd_even_flag" field in the key_stream_message() and shall use alternating table IDs to indicate the change in the ksm_table(). The correct usage of the table ID by the network and hardware section filtering in the receiving device may be used to limit local processing requirements.

It is advised to use a filter depth of 16 bytes or more (possibly even the maximal depth of 20 bytes as explained in ETSI ETR 289). The filter should incorporate at least the table ID and the odd_even_flag field from the key_stream_message.

Table 112 – KSM table

RI stream	Length	Type
ksm_table() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
DVB_reserved	1	bslbf
ISO_reserved	2	bslbf
CA_section_length	12	uimsbf
key_stream_message()		
}		

table_id – KSM_TABLE_ID as defined in Clause A.13. Two values for the table_id field are reserved for transmission of KSM data. A change of these two table ID values signals that a change of KSM contents has occurred, as explained in ETSI ETR 289.

EXAMPLE The change of the content key from odd to even may be indicated by a change from, for example, 0x081 to 0x080. Another example would be to signal the alternating inclusion or omission of next key material.

section_syntax_indicator – This field shall always be set to '0'.

DVB_reserved – This term indicates that the field may be used in the future for DVB applications and therefore shall not be used for private applications.

ISO_reserved – The term "ISO_reserved" indicates that the value may be used in the future for ISO defined extensions and therefore is not be specified by DVB.

CA_section_length – This field specifies the number of bytes that follow the section_length field up to the end of the section. The section shall not exceed 4093 so that the entire section has a maximum length of 4096 bytes.

key_stream_message() – Structure as defined in 7.2.

14.4 Delivery of rights management data in DVB T/C/S systems

14.4.1 General

DVB T/C/S systems use a broadcast channel for the delivery of rights management data. As an option, DVB T/C/S systems may use an interactivity channel for the delivery of rights management data. The use of broadcast channels and interactivity channels is specified in the next two subclauses.

14.4.2 Delivery of ICROs in DVB T/C/S systems over interactivity channel

Optionally, a DVB T/C/S system according to this standard may support the delivery of rights objects over an interactivity channel. The following two options are possible.

- When an interactivity channel is used in a bi-directional mode to deliver rights objects, all aspects of rights object delivery are governed by OMA DRM 2.0 specifications; see OMA-ERP-DRM-V2_0. Rights objects delivered over an interactivity channel are referred to as interactivity channel rights objects (ICROs).
- When an interactivity channel is used in a one-directional, downstream mode to deliver rights objects, the IP_linkage_descriptor (see 14.6.4.12) is used to signal that BCROs (8.4.2) are distributed in push mode *via the IP channel* to the device. See B.6.3 for details.

The interactivity channel used may be the Internet.

14.4.3 Delivery of BCROs in DVB T/C/S systems over broadcast channel

The private section as defined in ISO/IEC 13818-1 has a short and a long version. In a DVB T/C/S network, a short version of the private section is used to carry BCROs (see 8.4.2) as payload, as defined in Table 113. Hardware section filtering by the receiving device can be used to limit local processing requirements.

It is advised to use a filter depth of 16 bytes or more (possibly even the maximal depth of 20 bytes as explained in ETSI ETR 289). The filter should incorporate at least the table ID and the "address" field from the bcro_message.

The "linkage_descriptor" as specified in 14.6.4.11 may be used to signal the presence of BCROs in another, non-in-band channel, as suggested in B.6.3.

Table 113 – BCRO table

RI stream	Length	Type
bcro_table() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
private_section_length	12	uimsbf
bcro_message()		
}		

table_id – This field has the value of BCRO_TABLE_ID as defined in Table A.14.

section_syntax_indicator – The section_syntax_indicator is a 1-bit field that shall be set to '0'.

private_indicator – This is a 1-bit user definable flag that shall not be specified by ITU-T | ISO/IEC in the future.

private_section_length – This is a 12-bit field. It specifies the number of bytes of the section, starting immediately following the private_section_length field. The section shall not exceed 4093 so that the entire section has a maximum length of 4096 bytes.

bcro_message() – Structure BCRO() as defined in 8.4.2 with additional requirement that the size of the BCRO() shall never exceed 4093 bytes.

14.5 Delivery of registration data in DVB T/C/S systems

14.5.1 General

DVB T/C/S systems use a broadcast channel for the delivery of registration data. As an option, DVB T/C/S systems may use an interactivity channel for the delivery of registration data. The delivery of registration data over broadcast channels and interactivity channels is specified in 14.5.2 and 14.5.3, respectively.

14.5.2 Delivery of registration data in DVB T/C/S systems over an interactivity channel

Optionally, a DVB T/C/S system according to this standard may support the delivery of registration data over an interactivity channel. The following two options are possible.

- When an interactivity channel is used in a bi-directional mode to deliver registration data, all aspects of registration data delivery are governed by OMA DRM 2.0 specifications; see OMA-ERP-DRM-V2_0.However, see also 9.3.5 for the delivery of broadcast registration data using ROAP.
- When an interactivity channel is used in a one-directional, downstream mode to deliver registration data, the IP_linkage_descriptor (see 14.6.4.12) is used to signal that binary registration data (9.3.2) is distributed in push mode *via the IP channel* to the device. See B.6.3 for details.

The interactivity channel used may be the Internet.

14.5.3 Delivery of registration data in DVB T/C/S systems over a broadcast channel

In an MPEG2 TS, private data is carried in private sections of a maximum of 4kByte in length. Data that does not fit into one section can be carried in multiple sections. Corresponding to

this, a table header has been defined to carry the registration layer messages defined in 9.3. The messages of the registration layer shall be carried as defined in Table 114. The "linkage_descriptor" as specified in 14.6.4.11 may be used to signal the presence of data in another, non-in-band channel, as suggested in B.6.3.

Table 114 – Carrying registration layer messages via MPEG sections in T/C/S system

Message name	Table format	See subclause		Table ID will be[c]
device_registration_response()	RMT	9.3.2.7.2	[a]	DEVICE_REGISTRATION_TABLE_ID
re_register_msg()	RMT	9.3.2.7.3	[a]	RE_REGISTRATION_TABLE_ID
update_ri_certificate_msg()	RMT	9.3.2.7.4	[a]	UPDATE_RI_CERTIFICATE_TABLE_ID
update_drmtime_msg()	RMT	9.3.2.7.5	[a]	UPDATE_DRMTIME_TABLE_ID
update_contact_number_msg()	RMT	9.3.2.7.6	[a]	UPDATE_CONTACT_NUMBER_TABLE_ID
domain_registration_response()	RMT	9.3.3.5.1	[a]	DOMAIN_REGISTRATION_TABLE_ID
domain_update_response()	RMT	9.3.3.5.2	[a]	DOMAIN_UPDATE_TABLE_ID
join_domain_msg()	RMT	9.3.3.5.3	[a]	JOIN_DOMAIN_TABLE_ID
leave_domain_msg()	RMT	9.3.3.5.4	[a]	LEAVE_DOMAIN_TABLE_ID
token_delivery_response()	RMT	9.3.4.4	[b]	TOKEN_DELIVERY_TABLE_ID

[a] The filter should incorporate at least the table ID and the udn() field from the particular message that is carried as payload.

[b] The filter should incorporate at least the table ID and the address() field from the token_delivery_response() message that is carried as payload.

[c] See Clause A.13 for an overview of all table id values.

The registration message table (RMT) header is specified in 14.5.4.

14.5.4 Registration message table

The private section as defined in ISO/IEC 13818-1 has a short and a long version. In a DVB T/C/S network, a long version of the private section with extensions is used to carry the registration messages from 9.3 as payload in one or multiple sections, as specified in Table 115.

Hardware section filtering can be used by the receiving device to limit local processing requirements. It is advised to use a filter depth of 16 bytes or more (possibly even the maximal depth of 20 bytes as explained in ETSI ETR 289).

Table 115 – Syntax of registration message table (RMT)

Field	Length	Type
registration_message_table() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
private_section_length	12	uimsbf
table_id_extension	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
if (table_id == TOKEN_DELIVERY_TABLE_ID) {		
device_identification	40	bslbf
else {		
device_identification	64	bslbf
}		
unique_device_number	64	
for (i=0; i < section_length-5; i++){		
registration_message_byte()	8	bslbf
}		
CRC_32	32	rpchof
}		

table_id – This field will carry a value according to Table 114 depending on the type of message this table will carry.

section_syntax_indicator – The section_syntax_indicator is a 1-bit field that shall be set to '1'.

private_indicator – This is a 1-bit user definable flag that shall not be specified by ITU-T | ISO/IEC in the future.

private_section_length – This field specifies the number of remaining bytes in the section immediately following the section_length field up to the end of the section. The value in this field shall not exceed 4093 (0xFFD).

table_id_extension – This is a 16-bit field. Its use and value are defined by the user.

section_number – This field gives the number of the section. The section_number of the first section in a private table shall be 0x0. The section_number shall be incremented by 1 with each additional section in this private table.

last_section_number – This field specifies the number of the last section (that is, the section with the highest section_number) of the private table of which this section is a part.

version_number – This field is the version number of the private_section. The version_number shall be incremented by 1 modulo 32 when a change in the information carried within the private_section occurs. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable private_section with the same Table ID and section_number.

current_next_indicator – This field, which when set to '1', indicates that the private_section sent is currently applicable. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable private_section. When the bit is set to '0', it indicates that the private_section sent is not yet applicable and shall be the next private_section with the same section_number and Table ID to become valid.

device_identification – This field is used to incorporate device identification for filtering of the MPEG2 private section. Depending on the value of the table ID, this field can carry different information, as given below.

- In the case where the table ID signals the carriage of the token_delivery_response() message (i.e. table_id == TOKEN_DELIVERY_TABLE_ID) this field will carry the address() field from the token_delivery_response() message (see 9.3.4.4) as payload.
- In all other cases (i.e. table_id <> TOKEN_DELIVERY_TABLE_ID; see Table 114) this field carries the 64 most significant bits of the longform_udn() of the corresponding registration layer message. The longform_udn() is specified in 9.3.2.6.2.

NOTE Since the longform_udn() does not completely fit in the device_identification field, the device may in rare cases be triggered for a message that is not intended for it. When triggered by the MPEG section filtering, the device should therefore check the longform_udn() in the registration layer message.

registration_message_byte() – This field represents one of the subsequent bytes of one of the messages as defined in Table 114. The value in the field table_id determines the actual message carried in the longform_registration_message_table().

If the messages exceed the section length, the next section shall be used to carry the subsequent remaining part of the respective registration data message.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in ISO/IEC 13818-1, Annex A, after processing the entire private section.

14.6 Signalling and service guide in DVB T/C/S systems

14.6.1 General

ISO/IEC 13818-1 specifies service information that allows receivers to automatically demultiplex and decode the various programmes within the multiplex. These basic tables (PAT, PMT and CAT) are referred to as PSI tables. ETSI EN 300 468 provides additional tables (SDT, EIT, NIT etc) and descriptors that are used to provide additional information to the user. These tables are referred to as SI tables.

This standard adds additional tables and descriptors that are used to convey the necessary information needed to decrypt and purchase services protected by the technology in this standard. The tables and descriptors offer means to

- identify and locate the key stream for the different elementary streams;
- identify and locate rights issuer services conveying rights objects and other RI messages;
- label services with additional IDs to allow the creation of the unique CID used within this standard to identify services and their key material;
- identify and locate additional information needed for purchases, such as service bundles, information on the COC and purchase items;

- provide an informative schedule for the times that rights objects and other RI messages will be broadcast for particular sets of devices on an RI service.

The tables and descriptors defined in this standard adhere to the rules set out by ETSI EN 300 468 and ISO/IEC 13818-1.

14.6.2 Signalling of encrypted services in DVB T/C/S systems

14.6.2.1 General

This subclause and the following subclauses define how the tables in 14.6.3 and descriptors in 14.6.4 shall be used together with existing PSI/SI tables to convey the additional information needed to locate and identify key streams and rights issuer services, etc.

14.6.2.2 Signalling of key stream messages

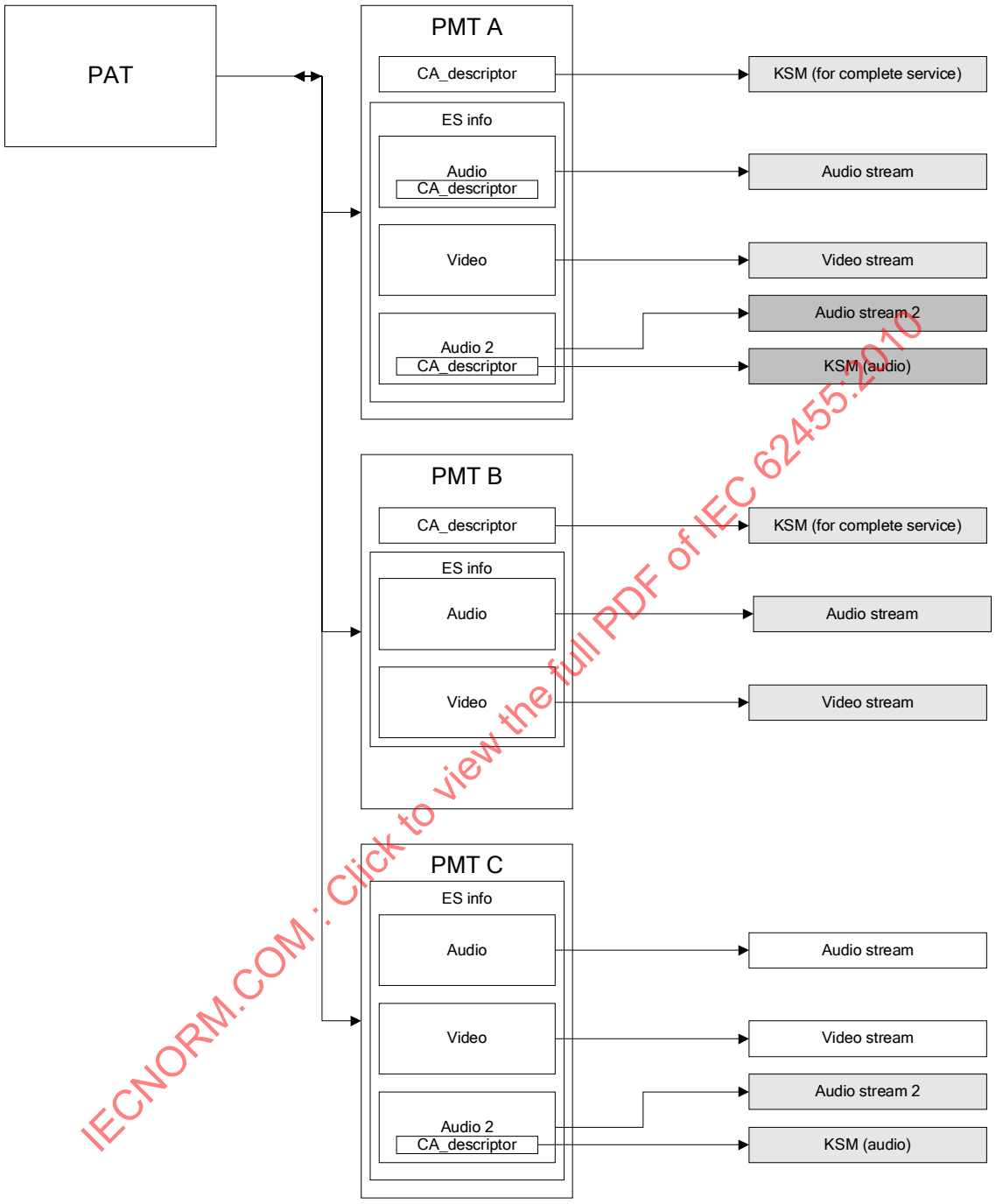


Figure 53 – Signalling of encrypted services and their associated key streams

Encrypted services are signalled by including a CA_descriptor (see ISO/IEC 13818-1, 2.6.16) in the PMT of that particular service. The CA_System_ID of the CA_descriptor shall be set to 18CRYPT_CAS_ID. The CAS_PID of the CA_descriptors specifies on which PID the KSM stream can be found. If the CA_descriptor is in the programme info descriptor loop of the PMT then the KSM stream is for all elementary streams listed in that PMT (see PMT B in Figure 53). If a CA_descriptor is used within the elementary stream descriptor loop of the PMT, then this overrides the settings for that particular elementary stream (see PMT A, Audio 2 in Figure 53). PMT C in Figure 53 shows the case where only one elementary stream of a service is scrambled. In this case, the CA_descriptor shall only be used within the elementary stream descriptor loop of that particular elementary stream.

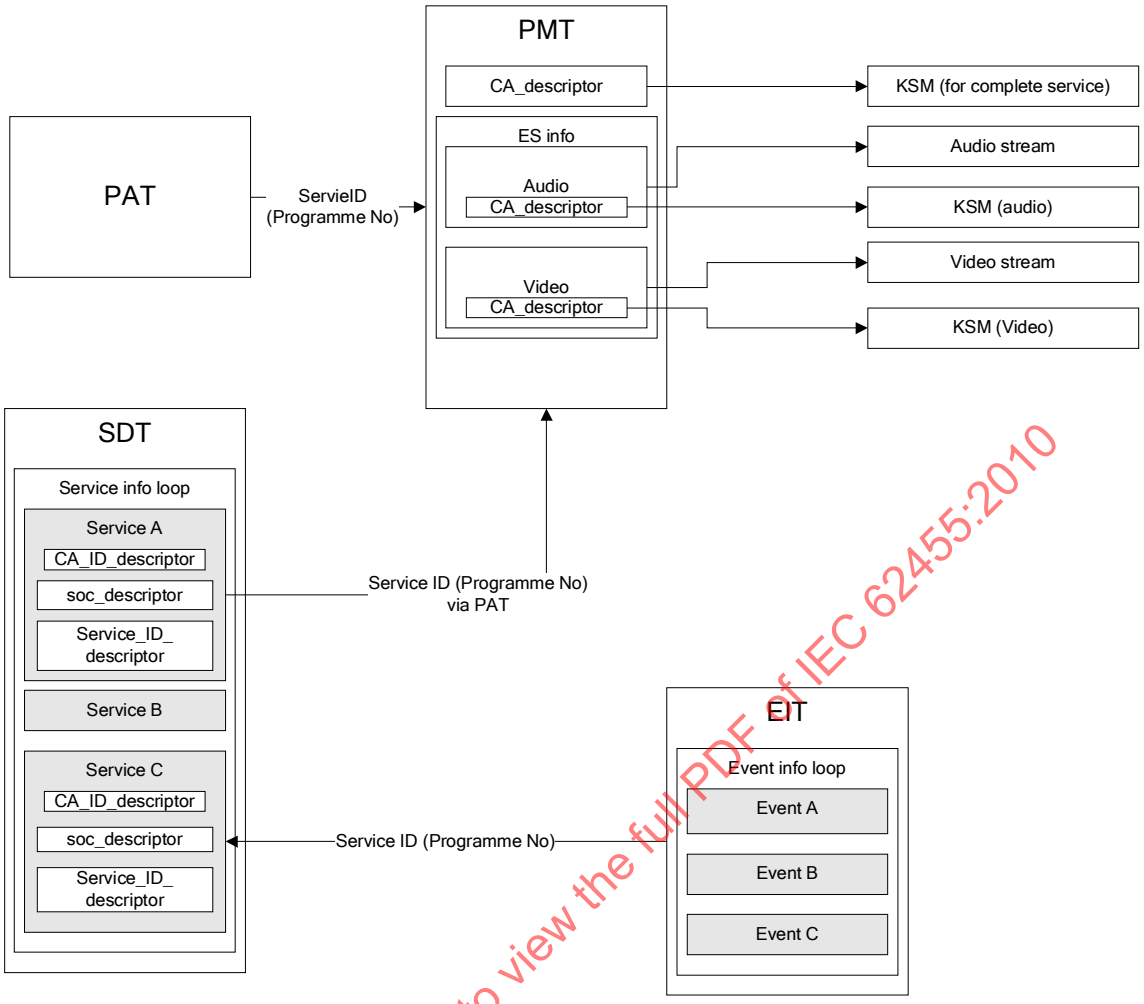


Figure 54 – Signalling of encrypted services in the SDT

Services encrypted using this standard shall be signalled in the SDT (ETSI EN 300 468, 5.2.3) by using the CA_identifier_descriptor (ETSI EN 300 468, 6.2.5) with the CA_System_ID set to 18CRYPT_CAS_ID and by using the service_ID_descriptor (see Table 121) as shown in Figure 54. The descriptor and the socID (a.k.a. original network ID) provide the necessary serviceBaseCID and SOC to create the CID that is used to link services, programmes and key material together.

14.6.2.3 Signalling of the rights issuer service in DVB T/C/S systems

The rights issuer service is signalled by setting the service_type of the service_descriptor (ETSI EN 300 468, 6.2.33) in the descriptor loop of the SDT to RI_SERVICE_TYPE (see Table 135) for that particular service. To identify the rights issuer a ri_id_descriptor (see Table 122) shall be present in that particular descriptor loop as shown in Figure 55. The actual RI stream containing BCRO tables etc is identified by the elementary stream listed in the PMT with the stream_type of RI_STEAM_TYPE (see Table 135). A particular rights issuer service might only be available on one transport stream while services offered by this rights issuer might be available on other transport streams. It is therefore up to the receiver to build up a database of rights issuers.

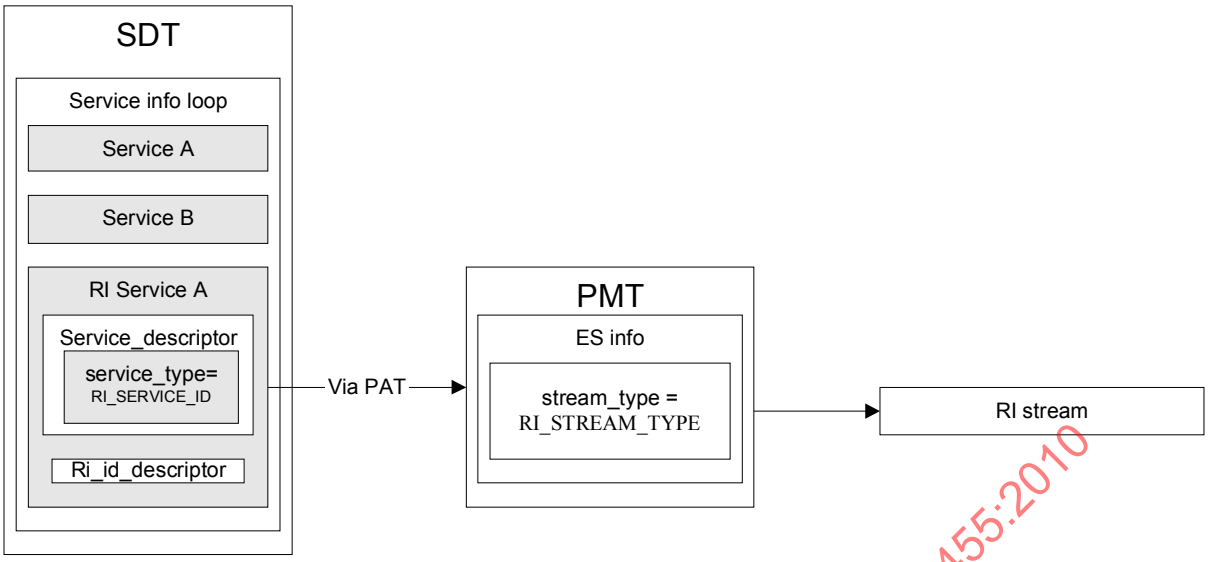


Figure 55 – Signalling of the rights issuer service in the SDT

In order to minimize filtering requirements in receivers the RI service may be split onto multiple streams, each stream addressing a different set of groups (see 5.5.2). The addressing shall be done by using a eurocrypt_addressing_descriptor in description of an elementary stream as shown in Figure 56.

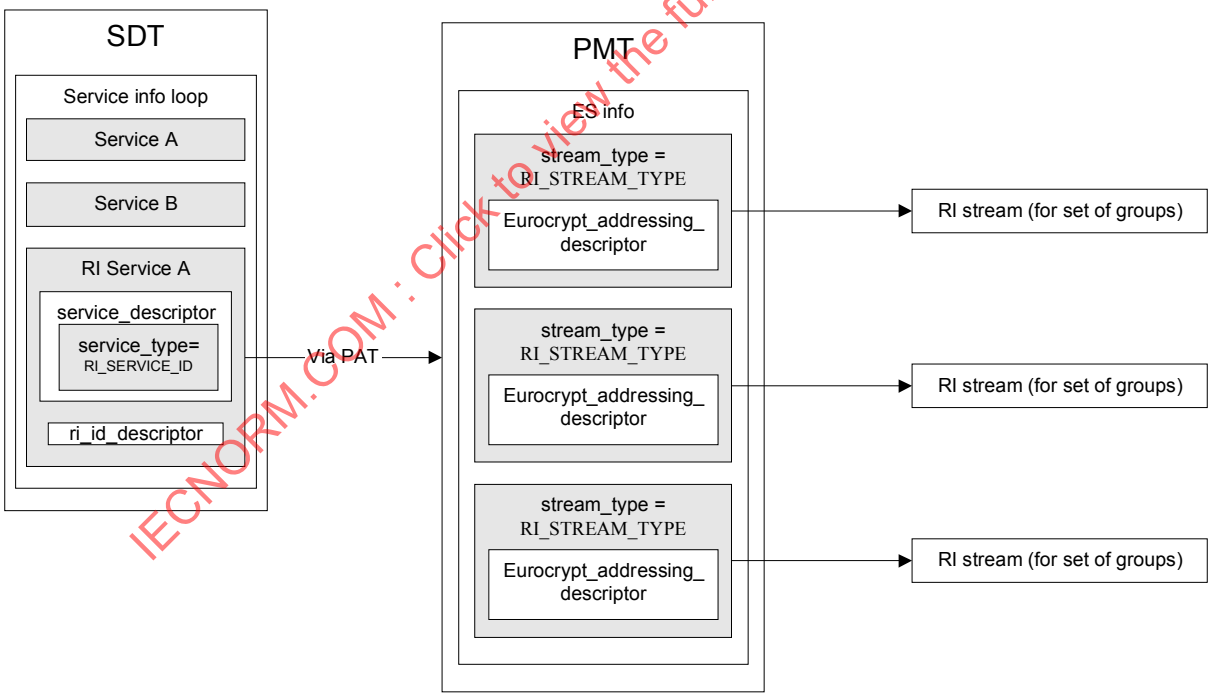


Figure 56 – Addressing of a rights issuer service

14.6.2.4 Signalling of an alternative RI service in DVB T/C/S systems

It is possible to signal an alternative/additional RI service on a different transport stream or delivered via an IP connection by using a linkage descriptor (see ETSI EN 300 468) with a linkage_type of RI_SERVICE_LINKAGE_ID or by using an IP linkage descriptor (see 14.6.4.12) with a linkage type of RI_SERVICE_LINKAGE_ID.

14.6.2.5 Signalling of purchase data in DVB T/C/S systems

There are three different ways of signalling purchase information (purchase channel, service bundles and purchase items).

The streams containing the purchase information tables (PCT 14.6.3.3, SBT 14.6.3.4 and PIT 14.6.3.5) may be signalled by using the purchase_info_location_descriptor (14.6.4.5) when listing an RI service in the SDT as shown in Figure 57. The streams referenced by the purchase_info_location_descriptor shall only contain tables for the rights issuer specified by the ri_id_descriptor for that particular RI service. It is, however, possible to have all three tables on the same stream by setting purchase_channel_pid, service_bundle_pid and purchases_item_pid in the purchase_info_location_descriptor to the same value.

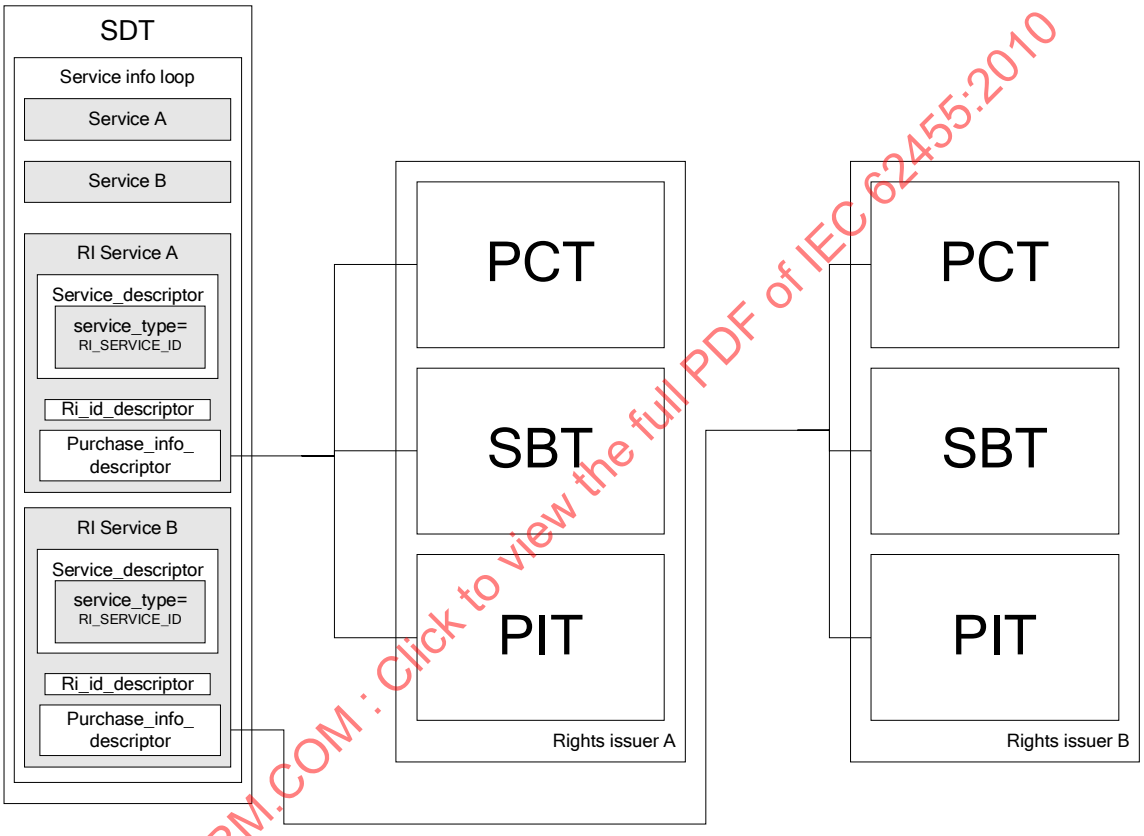


Figure 57 – Signalling of purchase information via the SDT

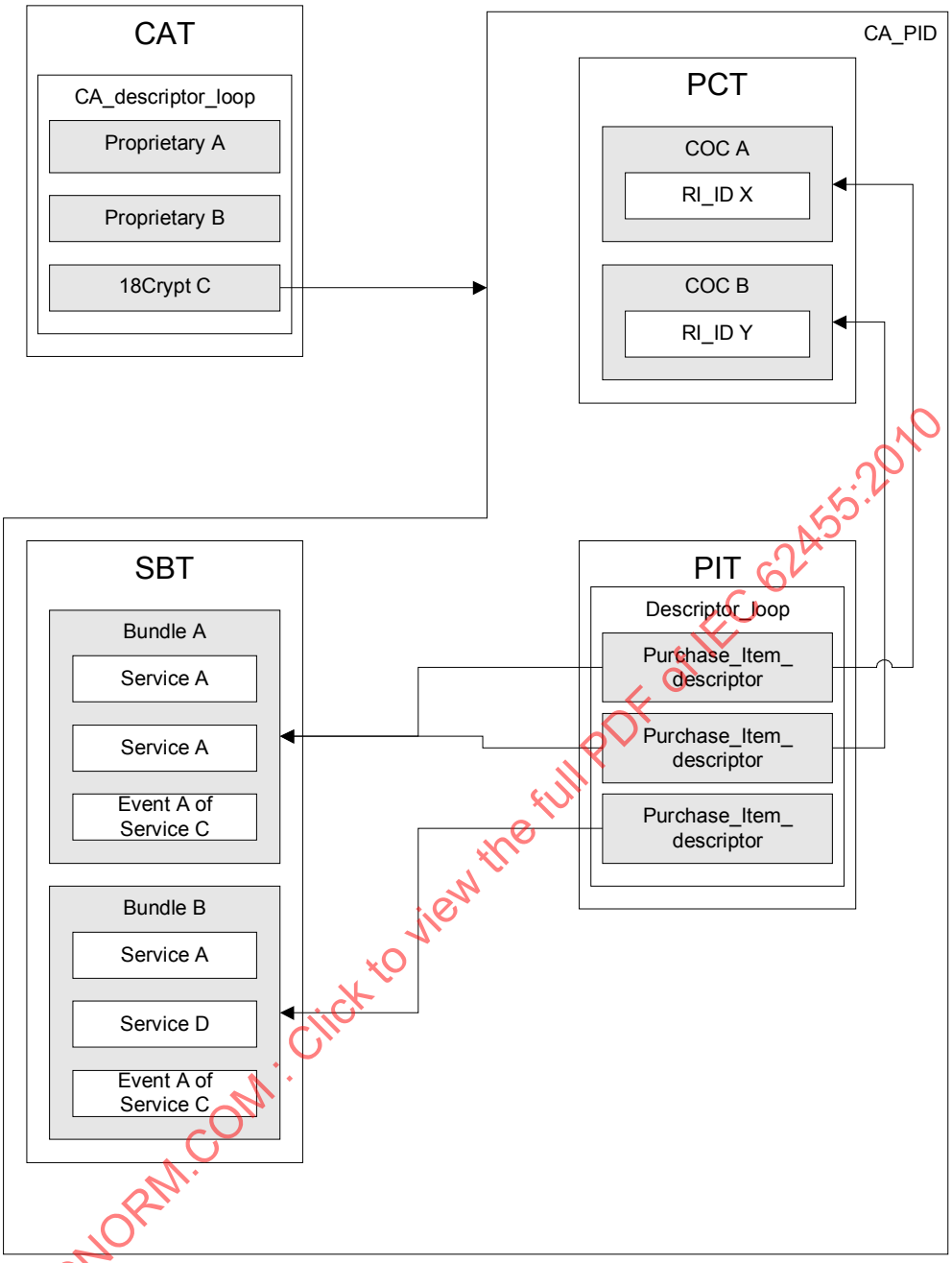


Figure 58 – Signalling of purchase information via the CA_descriptor in the CAT

Another way of signalling the purchase information is by using the CAT (see ISO/IEC 13818-1, 2.4.4.6) as shown in Figure 58. If the CAT contains a CA_descriptor with the CA_System_ID set to 18CRYPT_CAS_ID and the private data of the CA_descriptor does not contain a purchase_info_location_descriptor (see 14.6.4.5), then the CAS_PID of the CA_descriptors identifies the stream carrying the purchase information tables for all rights issuers offering services on this transport stream.

When the private data block of the CA_descriptor contains a purchase_info_location_descriptor, then the CA_PID of the CA_descriptor has no meaning and the PIDs specified in the purchase_info_location_descriptor shall be used to carry the purchase information (see Figure 59).

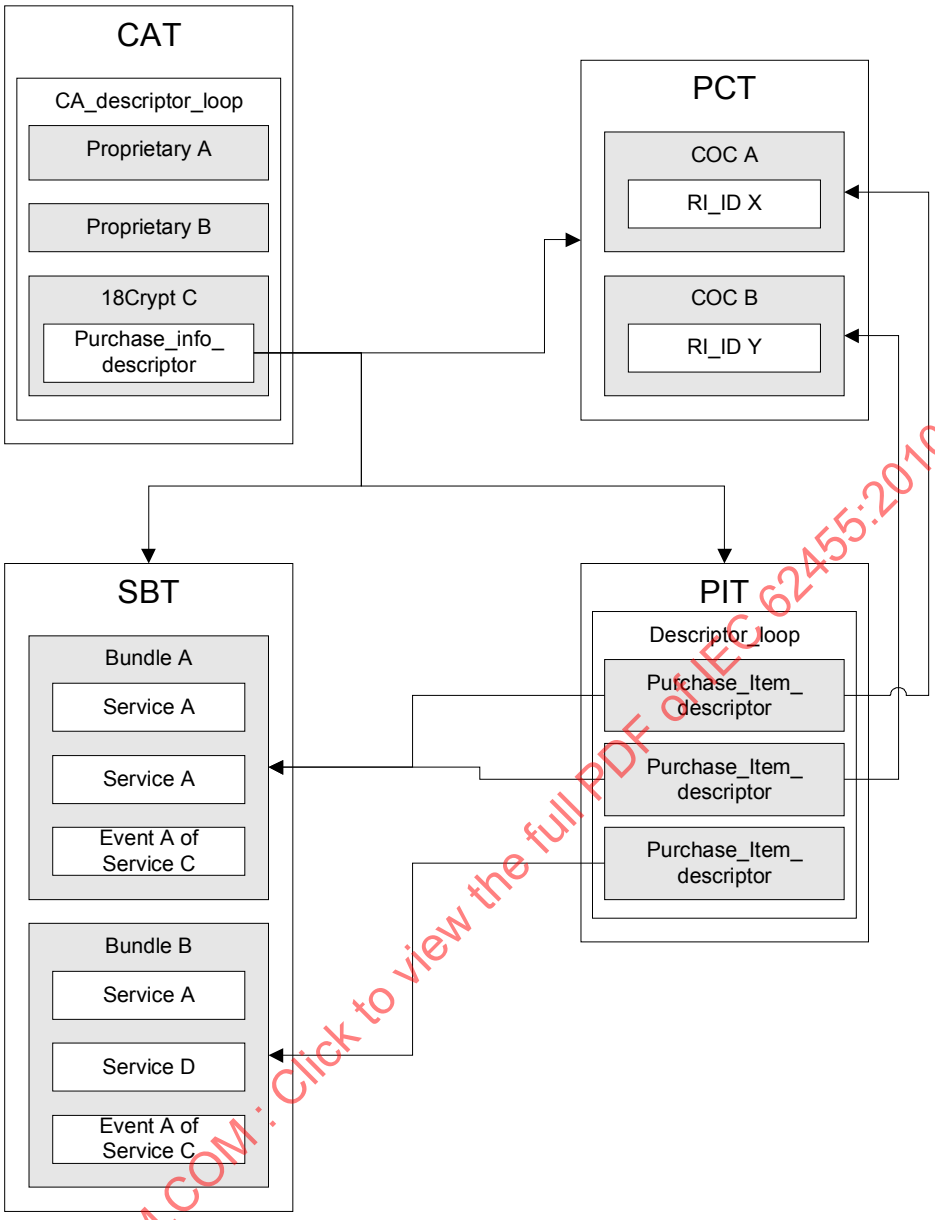


Figure 59 – Signalling of purchase information via the private data block of the CA_descriptor in the CAT

14.6.2.6 Relationship between purchase information and services

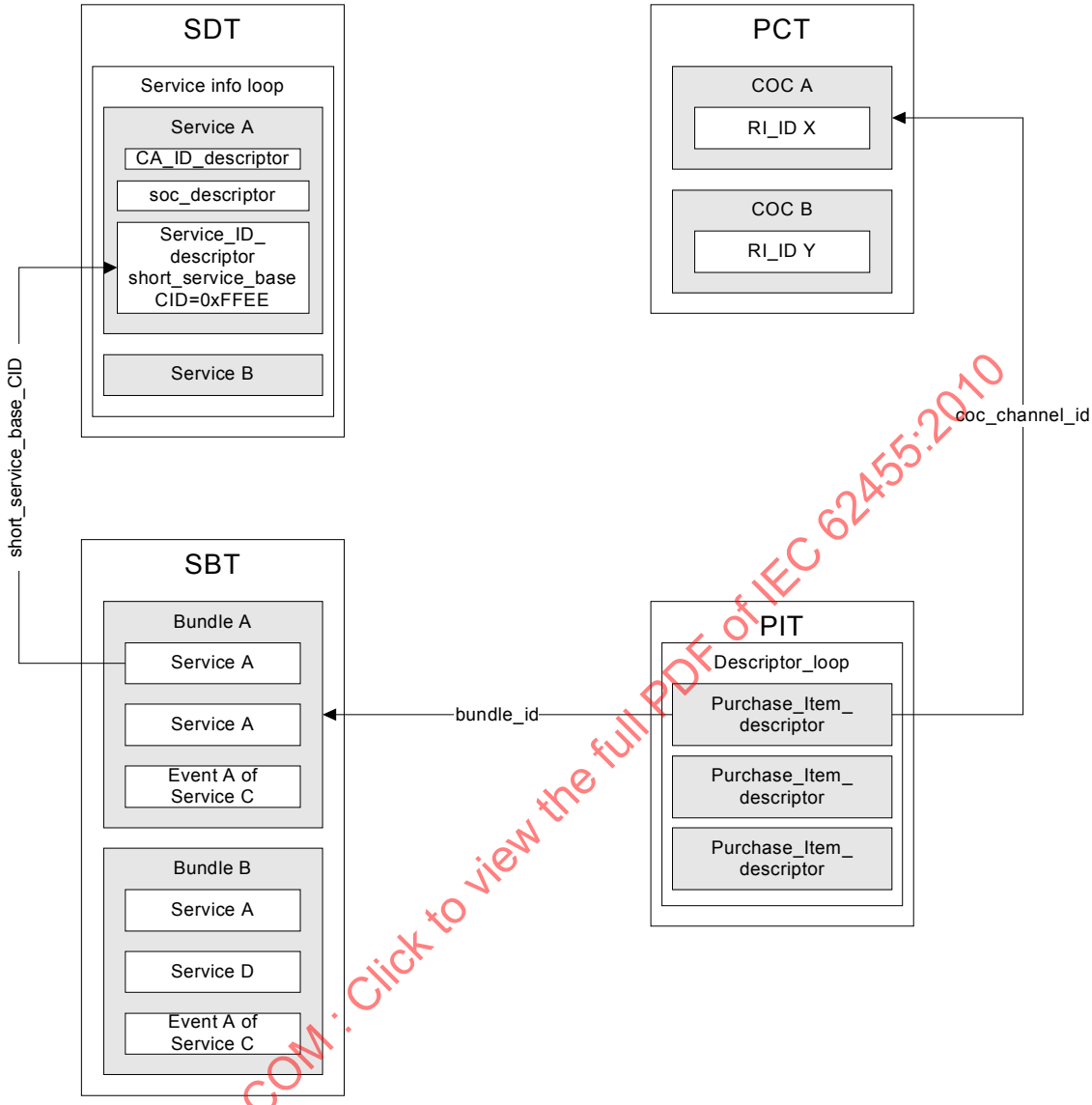


Figure 60 – Relationship between PCT, PIT, SBT and SDT

Figure 60 shows the relationship between the different tables for purchase information (PCT, SBT and PIT) and the SDT. It is, however, also possible to directly add pricing information to the service description or event description and avoid the use of SBT and PIT tables altogether as shown in Figure 61.

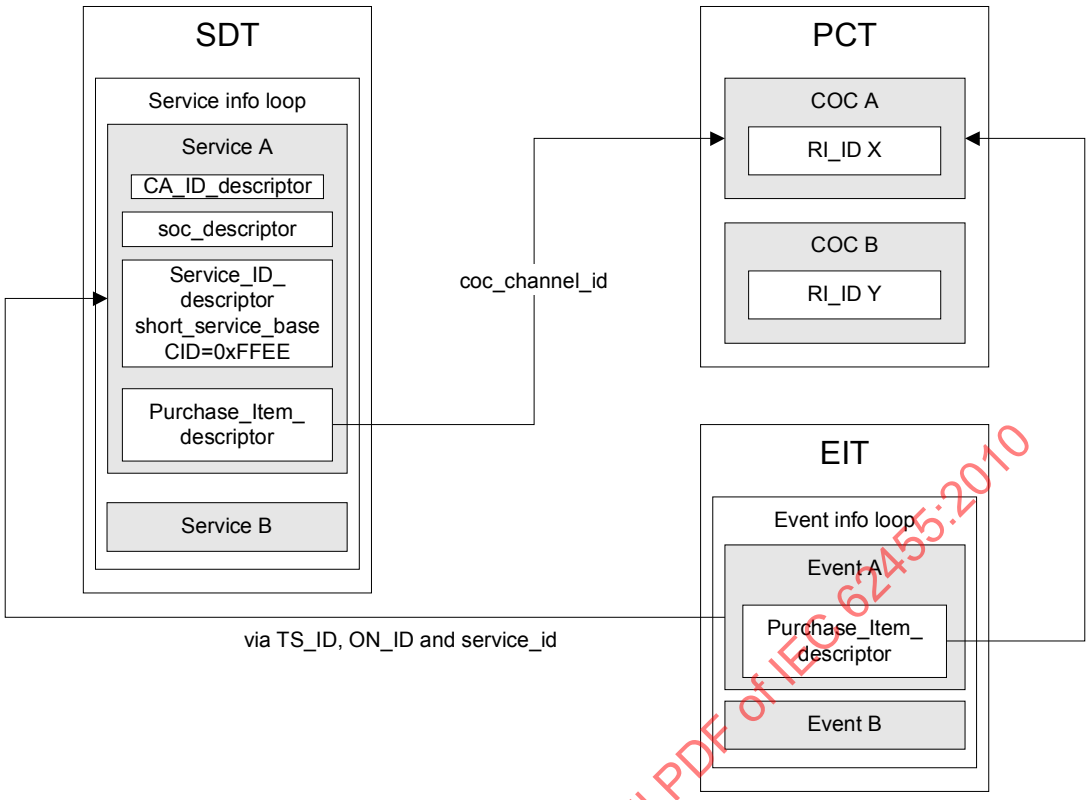


Figure 61 – Alternative usage of the purchase_item_descriptor in the SDT and EIT

14.6.2.7 Scheduling of rights issuer services

An informative schedule for right issuer services may be provided by the rights issuer. If the information is broadcast, devices may use it, for example, to save power by switching off at times when no messages are scheduled to be broadcast for the device.

The RI service is signalled as specified in 14.6.2.3. In addition, schedule information is carried in the EIT tables. Events that refer to schedule blocks in RI services

- shall refer to the RI service using the TS_ID, ON_ID and service_id in the usual way;
- shall contain start and end dates and times in the usual way;
- shall contain a EuroCrypt addressing descriptor (see 14.6.4.8), indicating the range of devices that will be addressed during this schedule block.

Rights Issuers should make reasonable efforts to meet the advertised schedule and shall note that failing to do so can lead to inefficient operation of the system and possible breaks in service to users.

It is RECOMMENDED that devices that make use of schedule information make some allowance for possible inaccuracies in the schedule by, for example, starting reception of data shortly before the advertised start time and continuing to receive data for a short period after the advertised end time.

14.6.3 SI tables

14.6.3.1 General

The SI tables defined in this subclause shall be segmented into one ore more sections before being inserted into TS packets.

These syntactic structures conform to the private section syntax defined in ISO/IEC 13818-1.

14.6.3.2 Table IDs

Clause A.13 lists the tables IDs that shall be used for the SI-tables defined in this standard.

14.6.3.3 Purchase channel table

The purchase channel table (PCT) in Table 116 conveys information regarding the different customer operation centres (see 5.1). The PCT is the equivalent to the XML PurchaseChannelType defined in ETSI TS 102 471. One coc channel loop can only be 4081 bytes long. Should the description of a COC (purchase channel) require more data, it can be transmitted in multiple sections with a different section_number. The COC is identified by the coc_channel_id in the loop. Information transmitted in multiple sections shall be treated as a concatenation of the different fields if these fields are of variable length. For example, if the purchase channel is split over two sections and in both sections the coc_purchase_url is present, for instance, in section 1 the coc_purchase_url field is set to http://www.MyPurchaseChannel.com and in section 2 it is set to '/purchase?coc=1234&lang=en' then the resulting coc_purchase_url is http://www.MyPurchaseChannel.com/purchase?coc=1234&lang=en.

For fields of fixed length, the content of the last section contributing to this one purchase channel shall be used. It is recommended however that the information of fixed size fields is the same in all sections contributing to this purchase channel.

Descriptors that this standard specifies for use with the PCT are listed in 14.6.4.2.

Table 116 – Purchase channel table

Field	Length	Type
purchase_channel_table() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
purchase_channel_collection_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0; i<N; i++){		
coc_id_length	8	uimsbf
coc_id	coc_id_length*8	uimsbf
coc_channel_id	16	uimsbf
short_rights_issuer_id	16	uimsbf
coc_name_loop_length	8	uimsbf
for(j=0; j<N; j++){		
multilingual_text()		
}		
coc_rekeying_safety_window	16	uimsbf
coc_purchase_url_flag	1	bslbf
coc_portal_url_flag	1	bslbf

Field	Length	Type
coc_contact_info_flag	1	bslbf
ri_url_flag	1	bslbf
reserved_future_use	4	bslbf
if(coc_purchase_url_flag){		
coc_purchase_url_length	8	uimsbf
coc_purchase_url	8*coc_purchase_url_length	bslbf
}		
if(coc_portal_url_flag){		
coc_portal_url_length	8	uimsbf
coc_portal_url	8*coc_portal_url_length	bslbf
}		
if(ri_url_flag){		
ri_url_length	8	uimsbf
ri_url	8*ri_url_length	bslbf
}		
if(coc_contact_info_flag){		
contact_loop_length	8	uimsbf
for(j=0; j<N; j++){		
contact_type	4	uimsbf
reserved_future_use	4	
coc_contact_info_length	8	uimsbf
coc_contact_info	8*coc_contact_info_length	bslbf
}		
}		
descriptor_loop_length	8	uimsbf
for(j=0; j<N; j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

table_id – PURCHASE_CHANNEL_TABLE_ID as defined in Table A.14.

section_syntax_indicator – The section_syntax_indicator is a 1-bit field that shall be set to '1'.

section_length – This is a 12-bit field. It specifies the number of bytes of the section, starting immediately following the section_length field. The section shall not exceed 4093 so that the entire section has a maximum length of 4096 bytes.

purchase_channel_collection_id – This is a 16-bit field that serves as a label to identify this particular collection of purchase channels.

version_number – This 5-bit field is the version_number of the sub_table. The version number shall be incremented by 1 when a change in the information carried in the sub_table occurs. When it reaches 31, it wraps around to 0. When the current_next_indicator is set to '1',

then the version_number shall be that of the currently applicable sub_table defined by the table_id and the purchase_channel_collection_id. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable sub_table defined by the table_id and purchase_channel_collection_id.

current_next_indicator – This 1-bit indicator, when set to '1' indicates that the sub_table is the currently applicable sub_table. When set to '0', it indicates that the sub_table sent is not yet applicable and shall be the next sub_table to be valid.

section_number – This 8-bit field gives the number of the section. The section_number of the first section in the sub_table shall be "0x00". The section_number shall be incremented by 1 with each additional section with the same table_id and purchase_channel_collection_id.

last_section_number – This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the sub_table of which this section is part.

coc_id_length – This is an 8-bit field that specifies the length in bytes of the following coc_id.

coc_id – This is a variable length field of coc_id_length bytes. The field contains the globally coordinated ID of the customer operation centre as a string. The encoding of the text and the character set used is specified in ETSI EN 300 468, Annex A.

coc_channel_id – This is a 16-bit field and identifies the COC uniquely within the original network and provides a mapping to the coc_id listed above. References to the COC within the SI data are done using this short ID.

short_rights_issuer_id – This is a 16-bit field and references the rights issuer this COC is bound to by using the short_right_issuer_id. See ri_id_descriptor in 14.6.4.4.

coc_name_loop_length – This is an 8-bit field that specifies the length in bytes of the following loop up to but excluding the coc_rekeying_safety_window.

multilingual_text() – This structure provides the name of the COC in a specific language. The syntax of this structure is specified in Clause A.14.

coc_rekeying_safety_window – This is a 16-bit field specifying the coc re-keying safety window as specified in 12.2.2.8.

coc_purchase_url_flag – 1-bit flag indicating the presence of the coc_purchase_url block. When set to 1 the block is present.

coc_portal_url_flag – 1-bit flag indicating the presence of the coc_portal_url block. When set to 1 the block is present.

coc_contact_info_flag – 1-bit flag indicating the presence of the coc_contact_info block. When set to 1 the block is present.

ri_url_flag – 1-bit flag indicating the presence of the ri_url block. When set to 1 the block is present.

coc_purchase_url_length – This is an 8-bit field specifying the length in bytes of the following coc_purchase_url. This field is only present when the coc_purchase_url_flag is set to 1.

coc_purchase_url – Variable length string of size 'coc_purchase_url_length' bytes. See 12.5.3. This field is only present when the coc_purchase_url_flag is set to 1. The encoding of the text and the character set used is specified in ETSI EN 300 468, Annex A.

coc_portal_url_length – This is an 8-bit field specifying the length in bytes of the following coc_portal_url. This field is only present when the coc_portal_url_flag is set to 1.

coc_portal_url – Variable length string of size 'coc_portal_url_length' bytes. See 12.5.3. This field is only present when the coc_portal_url_flag is set to 1. The encoding of the text and the character set used is specified in ETSI EN 300 468, Annex A.

ri_url_length – This is an 8-bit field specifying the length in bytes of the following ri_url loop. This field is only present when the ri_url_flag is set to 1.

ri_url – Variable length string of size 'ri_url_length' bytes. The URL is used by service subscription management for requesting ROAP triggers or requesting registration data or BCROs to be broadcast. This field is only present when the ri_url_flag is set to 1. The encoding of the text and the character set used is specified in ETSI EN 300 468, Annex A.

contact_loop_length – This is an 8-bit field specifying the length in bytes of the following contact_loop. This field is only present when the coc_contact_flag is set to 1.

contact_type – This 4-bit field specifies the type of contact as listed in Table 81.

coc_contact_info_length – This 8-bit field specifies the length of the following coc_contact_info field.

coc_contact_info – Variable length field of the coc_contact_info_length bytes. The encoding of the text and the character set used is specified in ETSI EN 300 468, Annex A. The format of the text is given by the contact_type.

descriptor_loop_length – This is an 8-bit field that specifies the length in bytes of all following descriptor() structures.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in ISO/IEC 13818-1, Annex A, after processing the entire section.

14.6.3.4 Service bundle table

The service bundle table (SBT) in Table 117 conveys information about the different service bundles that can be purchased. The SBT is the equivalent of the XML ServiceBundleType fragment specified in ETSI TS 102 471. A service bundle is defined by a 16-bit ID, which should be unique within the scope of the current network. Every service bundle is bound to one rights issuer either via the short_ri_id in the table or via the use of the purchase_info_location_descriptor in the signalling of an RI service (see 14.6.2.4). A service bundle may contain

- services, which are identified via the short_service_base_CID;
- events, which are identified via the short_service_base_CID and the event_id as specified by the EIT;
- components, for example, a second audio track, identified via the short_service_base_CID, an optional event_id and the component_no as specified in the PMT of that service.

The following descriptors may be used to provide additional information on the bundle:

- content_descriptor (see ETSI EN 300 468), to identify the genre(s) of this bundle;
- parental_rating_descriptor (see ETSI EN 300 468), to specify a parental rating of the bundle;
- provider_name_descriptor (see 14.6.4.7), to provide a multilingual name of the provider of the bundle;

- info_url_descriptor (see 14.6.4.9) to specify a URL to a web page where additional information to this bundle can be found.

Table 117 – Service bundle table

Field	Length	Type
service_bundle_table() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
service_bundle_collection_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0; i<N; i++){		
bundle_id	16	uimsbf
bundle_name_loop_length	8	
for(j=0; j<N; j++){		
multilingual_text()		
}		
ri_id_flag	1	
bundle_description_flag	1	bslbf
reserved_future_use	2	bslbf
descriptor_loop_length	12	uimsbf
if(ri_id_flag){		
short_ri_id	16	uimsbf
}		
if(bundle_description_flag){		
bundle_description_loop_length	8	uimsbf
for(j=0; j<N; j++){		
multilingual_text()		
}		
}		
list_loop_length	8	uimsbf
for(j=0; j<N; j++){		
short_service_base_CID	16	uimsbf
event_list_flag	1	bslbf
component_list_length	7	uimsbf
for(k=0;k<component_list_length; k++){		
component_no	8	uimsbf
}		
if(event_list_flag){		

Field	Length	Type
event_list_length	8	uimsbf
for(k=0; k<event_list_len/2; k++){		
event_id	16	uimsbf
}		
}		
}		
for(j=0; j<N; j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

table_id – SERVICE_BUNDLE_TABLE_ID as defined in Table A.14.

section_syntax_indicator – The section_syntax_indicator is a 1-bit field that shall be set to '1'.

section_length – This is a 12-bit field. It specifies the number of bytes of the section, starting immediately following the section_length field. The section shall not exceed 4093 so that the entire section has a maximum length of 4096 bytes.

service_bundle_collection_id – This is a 16-bit field that serves as a label to identify this particular collection of service bundles.

version_number – This 5-bit field is the version_number of the sub_table. The version number shall be incremented by 1 when a change in the information carried in the sub_table occurs. When it reaches 31, it wraps around to 0. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable sub_table defined by the table_id and the service_bundle_collection_id. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable sub_table defined by the table_id and service_bundle_collection_id.

current_next_indicator – This 1-bit indicator, when set to '1' indicates that the sub_table is the currently applicable sub_table. When set to '0', it indicates that the sub_table sent is not yet applicable and shall be the next sub_table to be valid.

section_number – This 8-bit field gives the number of the section. The section_number of the first section in the sub_table shall be "0x00". The section_number shall be incremented by 1 with each additional section with the same table_id and service_bundle_collection_id.

last_section_number – This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the sub_table of which this section is part.

bundle_id – This 16-bit field serves as a label to identify this service bundle. This ID is used in the purchase_item_descriptor to identify the service bundle. The ID '0' is reserved and shall not be used in this table.

bundle_name_loop_length – 8-bit field specifying the total length in bytes of the following multilingual bundle names.

multilingual_text() – This structure provides the bundle name in a specific language. The syntax of this structure is specified in Clause A.14.

ri_id_flag – 1-bit flag indicating the presence of the short_ri_id field.

bundle_decription_flag – 1-bit flag indicating the presence of the bundle description block.

descriptor_loop_length – This is a 12-bit field specifying the length of the descriptor loop in bytes that follows the list loop.

short_ri_id – This is a 16-bit field. When present it links the service bundle to the rights issuer identified by this short ID. If this table is carried on a stream containing purchase information for multiple rights issuers, this field shall be present.

bundle_description_loop_length – This 8-bit field specifies the length of byte of the following multilingual bundle descriptions.

multilingual_text() – This structure provides the bundle description in a specific language. The syntax of this structure is specified in Clause A.14.

list_loop_length – This 8-bit field specifies the total length in bytes of the following bundle definition up to but excluding the descriptor loop.

short_service_base_CID – This is a 16-bit field identifying a service as part of this service bundle via the short_service_base_CID as specified by the service_id_descriptor in the SDT. Should either one or more event_id fields or one or more component_no be present, then only the listed events/components are part of the service bundle and not the complete service itself.

event_list_flag – 1-bit field indicating the presence of the event_list block.

component_list_length – 7-bit fields specifying the total length of the following component_no list in bytes.

component_no – This 8-bit field specifies a particular component (elementary stream) of the service as specified by the stream_identifier_descriptor in the ES loop of the PMT. If no stream_identifier_descriptors are used in the PMT, the component_no specifies the loop number in the PMT with 0 being the first elementary stream, 1 the second listed etc.

event_list_length – 8-bit field specifying the total length in bytes of the following event list.

event_id – This is a 16-bit field containing the ID of the event as used in the EIT.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in ISO/IEC 13818-1, Annex A, after processing the entire section.

14.6.3.5 Purchase item table

The purchase item table (Table 118) conveys information for purchase items and is the equivalent to the XML PurchaseType of the DVB-CBMS ESG specification. The purchase items itself are described using the purchase_item_descriptor (see 14.6.4.6) in the descriptor loop of this table.

Table 118 – Purchase item table

Field	Length	Type
purchase_item_table() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
purchase_item_collection_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0; i<N; i++){		
descriptor()		
}		
CRC_32	32	rpchof
}		

table_id – PURCHASE_ITEM_TABLE_ID as defined in Table A.14.

section_syntax_indicator – The section_syntax_indicator is a 1-bit field that shall be set to '1'.

section_length – This is a 12-bit field. It specifies the number of bytes of the section, starting immediately following the section_length field. The section shall not exceed 4093 so that the entire section has a maximum length of 4096 bytes.

purchase_item_collection_id – This is a 16-bit field that serves as a label to identify this particular collection of purchase items.

version_number – This 5-bit field is the version_number of the sub_table. The version number shall be incremented by 1 when a change in the information carried in the sub_table occurs. When it reaches 31, it wraps around to 0. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable sub_table defined by the table_id and the purchase_item_collection_id. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable sub_table defined by the table_id and purchase_item_collection_id.

current_next_indicator – This 1-bit indicator, when set to '1' indicates that the sub_table is the currently applicable sub_table. When set to '0', it indicates that the sub_table sent is not yet applicable and shall be the next sub_table to be valid.

section_number – This 8-bit field gives the number of the section. The section_number of the first section in the sub_table shall be "0x00". The section_number shall be incremented by 1 with each additional section with the same table_id and purchase_item_collection_id.

last_section_number – This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the sub_table of which this section is part.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in ISO/IEC 13818-1, Annex A, after processing the entire section.

14.6.4 SI descriptors

14.6.4.1 Private data descriptor

The private data specifier descriptor (PDSD) indicating the start of a scope that contains 18Crypt private descriptors and/or user data shall use the private_data_specifier 18CRYPT_PRIVATE_DATA_ID. The value of this ID is assigned by DVB; see ETSI ETR 162.

The PDSD shall be in accordance with DVB requirements.

Table 119 – Private descriptor tags used for 18Crypt

Descriptor tag	Value
SERVICE_ID_DESCRIPTOR_TAG	0xA1
RI_ID_DESCRIPTOR_TAG	0xA2
PURCHASE_INFO_LOCATION_DESCRIPTOR_TAG	0xA3
PURCHASE_ITEM_DESCRIPTOR_TAG	0xA4
PROVIDER_NAME_DESCRIPTOR_TAG	0xA5
EUROCRYPT_ADDRESSING_DESCRIPTOR_TAG	0xA6
INFO_URL_DESCRIPTOR_TAG	0xA7
KEY_URL_DESCRIPTOR_TAG	0xA8
IP_LINKAGE_DESCRIPTOR_TAG	0xA9

Table 119 lists all descriptor tags used by 18Crypt in DVB-T/C/S networks. In order to use the following descriptors in a particular table, a PDSD with the private_data_specifier set to 18CRYPT_PRIVATE_DATA_ID has to be signalled in that table before the actual descriptor is signalled.

14.6.4.2 Descriptor locations

Table 120 lists which descriptors can be used in which table. The table covers the use of existing DVB-SI descriptors in new tables defined by this standard, new descriptors used in existing PSI/SI tables as well as new descriptors in new tables.

Table 120 – Possible locations of descriptors

Descriptor	Tag	CAT	NIT	SDT	EIT	PMT	PCT	SBT	PIT
content_descriptor	0x54	-	-	-	OK	-	-	OK	-
telephone_descriptor	0x57	-	-	OK	OK	-	OK	-	-
service_id_descriptor	0xA1	-	-	OK	-	-	-	-	-
ri_id_descriptor	0xA2	-	-	OK	-	-	-	-	-
purchase_info_location_descriptor	0xA3	OK [a]	-	OK [b]	-	-	-	-	-
purchase_item_descriptor	0xA4	-	-	OK	OK	-	-	-	OK
provider_name_descriptor	0xA5	-	-	-	-	-	-	OK	-
eurocrypt_addressing_descriptor	0xA6	-	-	-	OK [c]	OK [d]	-	-	-
info_url_descriptor	0xA7	-	OK [e]	-	-	-	-	OK	-
key_url_descriptor	0xA8	-	OK [f]	-	-	-	-	-	-
IP_linkage_descriptor	0xA9	-	-	OK	-	-	-	-	-

[a] In the private data block of the CA_descriptor.

[b] When signalling a right issuer service.

[c] Only when EIT references a rights issuer service.

[d] Only when stream_type is set to RI_STREAM_TYPE (see Table 135).

[e] Signalling the socInfoURL.

[f] Signalling the socKeyURL.

14.6.4.3 Service ID descriptor

The service_id_descriptor is used to signal the serviceBaseCID of a service and to provide a mapping to a shorter ID. This 'short_service_base_CID' is used to identify this service within the original network and should be unique within the scope of the original network (see original_network_id in the NIT).

Table 121 – Service_ID_descriptor

Field	Length	Type
service_ID_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0;i<descriptor_length-2;i++){		
serviceBaseCID_char	8	uimsbf
}		
short_service_base_CID	16	uimsbf
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to SERVICE_ID_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

serviceBaseCID_char – This is an 8-bit field. A string of 'char' fields specifies the serviceBaseCID. The length of the string is descriptor_length-2 bytes.

short_service_base_CID – 16-bit field specifying the 'short' serviceBaseCID. This short ID shall be unique within the scope of current network. The short serviceBaseCID is used to reference the serviceBaseCID in PSI tables.

14.6.4.4 Rights issuer ID descriptor

The rights issuer ID descriptor is used in the descriptor loop of the SDT when signalling a rights issuer service. It provides for a mapping of a short 16-bit long RI ID to the 160-bit RI ID. The short_RI_ID shall be unique within the scope of the original network.

Table 122 – Right issuer ID descriptor

Field	Length	Type
ri_id_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
rights_issuer_id	160	bslbf
short_RI_ID	16	uimsbf
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to RI_ID_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

rights_issuer_id – This is a 160-bit field containing the ID of the rights issuer. This is the 160-bit SHA-1 hash of the public key of the RI. See X509SPKIHash in OMA-TS-DRM-DRM-V2_0.

short_RI_ID – This is a 16-bit field that uniquely identifies the rights issuer within the scope of the original network. References to a specific rights issuer are done via the short_RI_ID that is mapped to the 160-bit rights_issuer_id via this descriptor.

14.6.4.5 Purchase info location descriptor

The purchase info location descriptor conveys information on where to find the purchase channel tables, the service bundle tables and the purchase item tables. The purchase info location descriptor can either be used in signalling of an RI service in the SDT or in the private data block of the CA_descriptor in the CAT. If the purchase info location descriptor is used as part of the signalling of an RI service, the referenced streams for the PCT, SBT and PIT are all bound to that particular rights issuer. The structure of the purchase info location descriptor is specified in Table 123.

Table 123 – Purchase info location descriptor

Field	Length	Type
purchase_info_location_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
purchase_channel_flag	1	bslbf
service_bundle_flag	1	bslbf
purchase_item_flag	1	bslbf
if((purchase_channel_flag |		
service_bundle_flag |		
purchase_item_flag) == 0){		
purchase_info_pid	13	uimsbf
}else{		
if(purchase_channel_flag){		
purchase_channel_pid	13	uimsbf
}else{		
reserved	5	bslbf
}		
if(service_bundle_flag){		
reserved_future_use	3	bslbf
service_bundle_pid	13	uimsbf
}		
if(purchase_item_flag){		
reserved_future_use	3	bslbf
purchase_item_pid	13	uimsbf
}		
}		
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to PURCHASE_INFO_LOCATION_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

purchase_channel_flag – 1-bit flag indicating the presence of the purchase_channel_pid field.

service_bundle_flag – 1-bit flag indicating the presence of the service_bundle_pid field.

purchase_item_flag – 1-bit flag indicating the presence of the purchase_item_pid field.

purchase_info_pid – This is a 13-bit field specifying the PID on which PCT, SBT and PIT tables are transmitted. When all three flags are set to '0', then all purchase info related tables (PCT, SBT, PIT) are transmitted on the same PID specified by this field.

purchase_channel_pid – This is a 13-bit specifying the PID on which purchase channel tables will be transmitted.

service_bundle_pid – This is a 13-bit specifying the PID on which service bundle channel tables will be transmitted.

purchase_item_pid – This is a 13-bit specifying the PID on which purchase item tables will be transmitted.

14.6.4.6 Purchase item descriptor

The purchase item descriptor conveys pricing information and restrictions for a particular service bundle, service or event. The purchase item descriptor can be used in the descriptor loop of the purchase item table. In this case, the purchase item described by this descriptor is linked to a service bundle identified by the bundle_id field. The structure of the purchase item descriptor is specified in Table 124.

If the purchase info descriptor is used within the descriptor loop of the SDT or in the descriptor loop of the EIT, then this purchase item is bound to the particular service or event respectively. The bundle_id shall be set to the reserved value of '0' in this case. As a descriptor can have a maximum length of only 256 bytes, information for one purchase item can be carried in multiple descriptors with the same purchase_id and coc_channel_id in the same table.

Information transmitted in multiple descriptors with the same purchase_id and coc_channel_id shall be treated as a concatenation of the different fields if these fields are of variable length. If a purchase item is split over two descriptor sections and in both sections, the English item description is present, the resulting English item description is the concatenation of both strings.

If multiple descriptors for the same purchase item carry pricing information in different currencies, for example, descriptor 1 has pricing information in EUR, and GBP and descriptor 2 as pricing information in USD then the complete purchase item has pricing information in EUR, GBP and USD. If the 2nd descriptor, however, has pricing info for GBP as well, then this overrides the information of the first descriptor.

For fields of fixed length, the content of the last section contributing to this one purchase channel shall be used. It is recommended however that the information of fixed size fields is the same in all descriptors contributing to this purchase item.

The purchase item descriptor can carry usage_constraint() objects that specify restriction of this purchase item to the user. For example, the time a subscription is valid (use of the Date-time constraint descriptor) or how often the content can be played (count_constraint_descriptor). These usage constraints are only for user information and shall not be used to enforce restrictions.

Table 124 – Purchase item descriptor

Field	Length	Type
purchase_item_descriptor() {		
descriptor_id	8	uimsbf
descriptor_length	8	uimsbf
purchase_id	32	uimsbf
bundle_id	16	uimsbf
coc_channel_id	16	uimsbf
purchase_item_name_loop_length	8	uimsbf
for(i=0; i<N; i++){		
multilingual_text()		
}		
purchase_item_version	8	uimsbf
item_description_flag	1	bslbf
subscription_type	3	bslbf
action_flag	1	bslbf
reserved_future_use	4	bslbf
if(item_description_flag){		
item_description_loop_length	8	uimsbf
for(i=0; i<N; i++){		
multilingual_text()		
}		
}		
price_loop_length	8	uimsbf
for(i=0; i<price_loop_length/6; i++){		
price_bcd	20	BCD
decimal_point_location	3	uimsbf
reserved_future_use	1	bslbf
currency_code	24	bslbf
}		
if(action_flag){		
action_loop_length	8	uimsbf
for(i=0; i<N; i++){		
action_type	7	uimsbf
constraint_flag	1	uimsbf
if(constraint_flag){		
constraint_loop_length	8	uimsbf
for(j=0; j<N; j++){		bslbf
constraint_descriptor()		
}		
}		
}		
}		
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to PURCHASE_ITEM_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

purchase_id – This is a 32-bit field and acts as a label of this particular purchase item. The ID should be unique within the scope of the customer operation centre (coc) this purchase item is bound to.

bundle_id – This is a 16-bit field identifying the service bundle this purchase item is bound to. If the bundle_id is 0 then the purchase item is bound to a particular service or event.

coc_channel_id – This is a 16-bit field identifying the COC channel this purchase item is bound to.

purchase_item_name_loop_length – 8-bit field specifying the length in bytes of the following multilingual purchase names.

multilingual_text() – This structure provides the purchase name in a specific language. The syntax of this structure is specified in Clause A.14.

purchase_item_version – This 8-bit field specifies the version of this purchase item.

item_description_flag – 1-bit flag indicating the presence of the item_description block.

subscription_type – 3-bit field specifying the type of subscription as shown in Table 125).

Table 125 – Subscription_type values

Subscription_type value	Description
1	Continues subscription
2	One-time subscription
3	Preview

usage_constraint_flag – 1-bit flag indicating the presence of the usage_constraint block. The usage constraint block is only for user information and shall not be used for real constraints. These are signalled in ROs or the KSM.

item_description_loop_length – This 8-bit field specifies the length in bytes of the following multilingual purchase item description up to but excluding the price_loop_length field.

multilingual_text() – This structure provides the purchase item description in a specific language. The syntax of this structure is specified in Clause A.14.

price_loop_length – This 8-bit field specifies the length in bytes of the following price loop up to but excluding the usage_constraint block.

The loop following the price_loop_length field contains the prices for the purchase item in different currencies. This loop shall not contain two prices with the same currency.

price_bcd – This is a 20-bit field containing five BCD (binary coded decimal) coded digits. The price of the purchase item is derived by inserting a decimal point starting from the left at the position indicated by the decimal_point_location+2. The smallest possible price (apart from 0) is 0,001 while the biggest possible price is 999 990 000,00.

EXAMPLE If the price_bcd field has the value 0x10952, the five digits are 10952. The different prices, which can be derived from this with different decimal_point_location values, are shown in Table 126.

Table 126 – Example price with different decimal point location values

Decimal_point_location value	Decimal point location (from the left)	Price
0	2	10,952
1	3	109,52
2	4	1 095,20
3	5	10 952,00
4	6	109 520,00
5	7	1 095 200,00
6	8	10 952 000,00
7	9	109 520 000,00

decimal_point_location – 3-bit field specifying the decimal point of the price minus 2. A value of 0 means the decimal point is at after the second digit from the left, a value of 1 means the decimal point is after the third digit from the left, etc.

currency_code – This is a 24-bit field containing the ISO/IEC 8859-1:1998 coded 3-letter currency code according to ISO 4217.

EXAMPLE The Euro (€) has the currency code 'EUR' which is coded as "0100 0101 0101 0101 0101 0010".

action_flag – 1-bit flag indicating the presence of the action block. The action block is only for user information and shall not be used for real constraints. These are signalled in ROs or the KSM.

action_type – 7-bit field specifying the action_type as defined in Table 36.

constraint_flag – 1-bit flag indicating the presence of the constraint block. The constraint block is only for user information and shall not be used for real constraints. These are signalled in ROs or the KSM.

constraint_loop_length – 8-bit fields specifying the length in bytes of the following usage constraint loop. Multiple usage constraints shall be treated as having an AND relationship. The format of the following constraint_descriptor is defined in 8.4.6. The usage constrained listed in this descriptor are for user information only and shall not be used to enforce usage constrained. This is solely done via the usage constraints in the rights objects or the KSMs.

14.6.4.7 Provider name descriptor

The provider name descriptor conveys the multilingual name of a provider. The structure of the provider name descriptor is specified in Table 127.

Table 127 – Provider name descriptor

Field	Length	Type
provider_name_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0; i<N; i++){		
multilingual_text()		
}		
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to PROVIDER_NAME_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

multilingual_text() – This structure provides the provider name in a specific language. The syntax of this structure is specified in Clause A.14.

14.6.4.8 Eurocrypt addressing descriptor

The eurocrypt addressing descriptor may be used

- in the elementary stream loop in a PMT of a rights issuer service. A rights issuer service can have multiple elementary streams of type RI_STREAM_TYPE (see Table 135). By using this descriptor, the streams can be associated to certain groups;
- in the event loop of an EIT to allow scheduling of RI messages to specific groups.

The structure of the eurocrypt addressing descriptor is specified in Table 128.

Table 128 – Eurocrypt addressing descriptor

Field	Length	Type
eurocrypt_addressing_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reserved_future_use	4	bslbf
group_size_flag	1	bslbf
address_mode	3	uimsbf
address	32	uimsbf
if(address_mode == 0x1 && group_size_flag == 0) { bit_access_mask }	256	bslbf
else if(address_mode==0x1 && group_size_flag==1) { bit_access_mask }	512	bslbf
else if (address_mode&0x6 == 0x2) { position_in_group }	8	uimsbf
}else if (address_mode == 0x4){		
domain_id_extension	6	bslbf
domain_generation	10	uimsbf
}else if(address_mode == 0x5){ group_mask }	32	bslbf
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to EUROCRYPT_ADDRESSING_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

group_size_flag – 1-bit field indicating the group size used. 0 – a maximum group size 256 is used, 1 – a maximum group size of 512 is used.

address_mode – 3-bit field indicating the addressing mode used by this descriptor. Table 129 lists all possible address modes.

Table 129 – Address_mode

Address_mode	Description
0x0	Addressing whole of unique group
0x1	Addressing of subscriber group using a bit_mask size of 256 or 512 bit depending on group_size_flag (subset of unique group)
0x2-0x3	Addressing of unique device
0x4	Addressing of OMA domain. Address field concatenated with the domain_id_extension will be the domain id in this case
0x5	Addressing set of groups by applying group_mask to address
0x6-0x7	Reserved

address – 4-byte group address. Each rights issuer has its own address space. If the group_size is 512 then the group address is made of the first 31 bit of the address field. If the descriptor is addressing a unique device in a group then the least significant bit of the address field is the most significant bit of the group position. If the address_mode is set to 0x4, the address field contains the first 32 bit of the short form domain_id.

bit_access_mask – If the descriptor addresses a subset of a unique group (address_mode 0x1) than the bit_access_mask defines to which devices in the group this descriptor is addressed to. Devices not listed in the bit_access_mask cannot decrypt the key material in this BCRO as zero message broadcast encryption is used for the encryption of the key material. The size of the bit_access_mask is given by the group_size_flag.

position_in_group – If the descriptor addresses a unique device then this field specifies the position of the unique device in the given subscriber group. If group_size_flag is 0 than the position in the group is directly given by the position_in_group field. If group_size_flag is 1 then 9 bits are used to identify the position in the group. If group_size_flag is 1 then the least significant bit (bit 0) from the address field is used as the 9th bit, the most significant bit. The real position in the group is then given by:

```
int real_position_in_group;
if(address_mode&0x6==0x2){
  if(group_size_flag == 0){
    //maximum size of 256 devices in group.
    real_position_in_group = position_in_group;
  }else{
    //maximum size of 512 devices in group;
    real_position_in_group = ((address&0x1)<<8)|position_in_group;
  }
}
```

domain_id_extension – The domain_id is given by the address field concatenated with the domain_id_extension to form a 38-bit id:

```
domain_id = (address<<6)|domain_id_extension
```

domain_generation – This 10-bit field specifies the generation of the domain.

group_mask – If the address_mode is set to 0x5 then this 32-bit field is used together with the 32-bit address field to specify the range of groups the descriptor is addressed to. A device, which is in the group with the address 'my_group', is addressed if the following statement is true:

address&group_mask == my_group&group_mask

14.6.4.9 Info URL descriptor

The info URL descriptor conveys a URL from which additional information can be obtained. The structure of the Info URL descriptor is specified in Table 130.

Table 130 – Info URL descriptor

Field	Length	Type
info_url_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0; i<descriptor_length; i++){		
url_char	8	uimsbf
}		
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to INFO_URL_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

url – Variable length field of descriptor_length bytes. The text representing the URL is encoded according to ISO/IEC 8859-1:1998.

14.6.4.10 Key URL descriptor

The key URL descriptor conveys a URL through which key material can be retrieved. The structure of the Info URL descriptor is specified in Table 131.

Table 131 – Key URL descriptor

Field	Length	Type
key_url_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0; i<descriptor_length; i++){		
url_char	8	uimsbf
}		
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to KEY_URL_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

url – Variable length field of descriptor_length bytes. The text representing the URL is encoded according to ISO/IEC 8859-1:1998.

14.6.4.11 Linkage descriptor

The linkage descriptor is defined in ETSI EN 300 468. This standard extends the linkage descriptor by defining the structure for a linkage descriptor (as shown in Table 132) for linkage_types as shown in Table 133.

Table 132 – Linkage descriptor

Field	Length	Type
linkage_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
service_id	16	uimsbf
linkage_type	8	uimsbf
if (linkage_type ==0xBB){		
short_RI_ID	16	uimsbf
}else{		bslbf
for (i=0; i<–N; i++){		
private_data_byte	8	bslbf
}		
}		
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to 0x4A.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

transport_stream_id – This is a 16-bit field that identifies the TS containing the information service indicated.

original_network_id – This 16-bit field gives the label identifying the network_id of the originating delivery system of the information service indicated.

service_id – This is a 16-bit field that uniquely identifies an information service within a TS. The service_id is the same as the program_number in the corresponding program_map_section.

linkage_type – This is an 8-bit field specifying the type of linkage. This standard only defines linkage_types as listed in Table 133. Other linkage types are defined in ETSI EN 300 468.

Table 133 – Linkage type coding

Linkage_type	Description
0x00-0x7F	Defined by other standards
0x80-0xBA	Reserved for future use (within the scope of 18CRYPT_PRIVATE_DATA_ID)
0xBB	RI service
0xBC	Purchase service
0xBD	EIT service
0xBE-0xFE	Reserved for future use (within the scope of 18CRYPT_PRIVATE_DATA_ID)
0xFF	Reserved for future use

short_RI_ID – This 16-bit field specifies the short version of the RI ID the referred to RI service belongs to. This field is only present when the linkage_type is 0xBB.

private_data_byte – This is an 8-bit field, the value of which is privately defined.

14.6.4.12 IP linkage descriptor

The IP linkage descriptor as defined in Table 134 is in its concept similar to the linkage descriptor. It is used to signal alternative or additional service. While the linkage descriptor refers to MPEG services, the IP linkage descriptor refers to services carried over an UDP/IP connection.

Table 134 – IP linkage descriptor

Field	Length	Type
IP_linkage_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reserved_future_use	7	uimsbf
IP_version	1	uimsbf
port	16	uimsbf
linkage_type	8	uimsbf
if(IP_version == 0){		
IP_address	32	uimsbf
}else{		
IP_address	128	uimsbf
}		
if (linkage_type == 0xBB){		
short_RI_ID	16	uimsbf
}else{		bslbf
for (i=0; i<N; i++){		
private_data_byte	8	bslbf
}		
}		
}		

descriptor_tag – This is an 8-bit field identifying the descriptor. The tag shall be set to IP_LINKAGE_DESCRIPTOR_TAG as defined in Table 119.

descriptor_length – This is an 8-bit field specifying the total number of bytes of the data portion of the descriptor following the byte defining the value of this field.

IP_version – 1-bit flag indicating the IP version used in for the IP_address field in this descriptor.

port – 16-bit field specifying the port on which the referred to service can be received.

IP_address – Field containing the IP address the referred service can be received on. The format and length of this field is specified by the IP_version field. If IP_version is '0' then the field specifies an IPv4 address and the length of the field is 32-bit. If IP_version is set to '1', then the field specifies an IPv6 address and the length is 128-bit.

linkage_type – This is an 8-bit field specifying the type of linkage. This standard only defines linkage_types as listed in Table 133. Other linkage types are defined in ETSI EN 300 468.

The services listed in Table 133 are normally carried in MPEG sections. In order to transmit these services over an IP connection the sections are carried in UDP packets. Every UDP packet shall contain exactly one MPEG section.

short_RI_ID – This 16-bit field specifies the short version of the RI ID the referred to RI service belongs to. This field is only present when the linkage_type is 0xBB.

private_data_byte – This is an 8-bit field, the value of which is privately defined.

14.7 User-defined identifiers used in DVB-SI tables

Table 135 – User defined IDs

ID	Value	Comment
RI_STREAM_TYPE	0xBB	Used in stream_type in PMT. The descriptor loop in the elementary stream loop for the rights issuer shall contain a private_data_specifier_descriptor with the private_data_specifier set to 18CRYPT_PRIVATE_DATA_ID to indicate the scope of the user defined values; see Table 2-34 – Stream type assignments in ISO/IEC 13818-1.
RI_SERVICE_TYPE	0xBB	Used in service_descriptor to signal a rights issuer service. A private_data_specifier_descriptor with the value of 18CRYPT_PRIVATE_DATA_ID shall be signalled in the descriptor loop of the SDT before the service_descriptor to identify the scope of the user define value.
RI_SERVICE_LINKAGE	0xBB	Used in a linkage descriptor or an IP linkage descriptor to signal the existence of an alternative/additional right issuer service.
PURCHASE_STREAM_TYPE	0xBC	Used in stream_type in PMT. A service with this stream type only carries purchase related tables such as PIT, PCT and SBT.
PURCHASE_SERVICE_TYPE	0xBC	Used in service_descriptor to signal a purchase service only containing purchase related tables. This kind of service is used together with a linkage descriptor in order to allow for purchase information to be delivered on a different transport.
PURCHASE_SERVICE_LINKAGE	0xBC	Used in a linkage descriptor or an IP linkage descriptor to signal the existence of a (for example, out-of-band) purchase service.
EIT_SERVICE_LINKAGE	0xBD	Used in a linkage descriptor or an IP linkage descriptor to signal the existence of a Service carrying EIT tables on a different transport stream (for example, on out-of-band transport stream).
EIT_STREAM_TYPE	0xBD	Used in stream_type in PMT A stream with this stream type only contains MPEG sections containing EIT tables.
EIT_SERVICE_TYPE	0xBD	Used in service_descriptor to signal an EIT service. This kind of service is used together with a linkage descriptor to carry EIT information on a different transport stream (for example, via out-of-band).

14.8 Scope of identifiers used in DVB-SI tables

The signalling for DVB-T/C/S systems introduces a couple of 'short' IDs that are mapped to longer IDs used throughout this standard. These short IDs are used to save bandwidth but while the long form IDs are unique, the short IDs are only unique within a certain scope. This subclause specifies the scope of the different short IDs used and the entity responsible for assigning these IDs.

short_service_base_CID – The short_service_base_CID is a 16-bit ID that maps to the variable length serviceBaseCID. The short_service_base_CID is only unique within the scope of the original network. A network could contain multiple transport streams from different original networks (see network_id and original_network_id in the NIT). Only the combination o original_network_id and short_service_base_CID makes this ID unique in a network. It is up to the original network operator to assign these IDs.

coc_channel_id – The coc_channel_id is a 16-bit ID that maps to the variable length coc_id. The coc_channel_id is only unique within the scope of the original network. A network could contain multiple transport streams from different original networks (see network_id and original_network_id in the NIT). Only the combination o original_network_id and coc_channel_id makes this ID unique in a network. It is up to the original network operator to assign these IDs.

short_RI_ID – The short_RI_ID is a 16-bit ID that maps to the 160-bit rights issuer ID. The short_RI_ID is only unique within the scope of the original network. A network could contain multiple transport streams from different original networks (see network_id and original_network_id in the NIT). Only the combination o original_network_id and short_RI_ID makes this ID unique in a network. It is up to the original network operator to assign these IDs.

purchase_id – This is a 16-bit ID and acts as a label of a particular purchase item. The ID shall be unique within the scope of a customer operation centre (coc).

bundle_id – This is a 16-bit ID identifying a service bundle. This ID shall be unique within the scope of one rights issuer.

14.9 Format of RI services over DVB-T/C/S systems

14.9.1 General

In DVB-T/C/S systems, rights management data is carried as specified in 14.4 and registration data is carried as specified in 14.5.

The RI service for DVB-H systems is specified in 13.7.3. The RI service for DVB-T/C/S systems is the same as the RI service for DVB-H systems, except for the differences specified in 14.9.2, 14.9.3 and 14.9.4.

14.9.2 RI stream packet format

In DVB-T/C/S systems, the RI services carry different payloads in the objects, which in this case are the MPEG2 private section tables RMT (see 14.5.3) and bcro_table (see 14.4.3). The objects themselves are put into RI packets as defined in 13.7.3. The RI packet format is defined in Table 110. The RI packets are divided by the MPEG2 TS into MPEG2 TS packets. There are therefore no changes in IP packet order (also see 14.9.4). In DVB T/C/S systems, there is no mapping to DVB-H time slice bursts.

14.9.3 Addressing of objects

The addressing of objects is done by filtering on the MPEG2 private header as specified in 14.5.3 and 14.4.3.

14.9.4 Mapping of messages to RI services and streams

Within a DVB-T/C/S network, devices discover streams using EPG and various SI/PSI tables. See Clause B.6 for more information.

A DVB-T/C/S system with the optional OOB broadcast channel shall support the linkage descriptor, and in this case, the RI services shall support getting RI services data from that channel by making the proper associations between the PIDs as identified in the PSI/SI data. In this case, the objects are carried in MPEG2 TS packets.

A DVB-T/C/S system with the optional interactivity channel shall support the IP_linkage descriptor, and, in this case, the RI services shall support getting RI services data from that channel by making the proper associations between the IP address and port number as identified in the PSI/SI data. In this case, the objects are carried in UDP packets.

15 Protection of MPEG2 TS-based IP systems

15.1 General

This clause specifies the use of specific protection technologies from Clauses 6, 7, 8, 9, 10 and 11 for IP-based broadcast systems that employ an MPEG2 transport stream for the encapsulation of the content. Clauses 4, 5 and 12 are general and apply to all systems. An

example of such a system is specified by the DVB-IPI specifications (ETSI EN 102 034 and Clause B.11 for general references).

15.2 Encapsulation of an MPEG2 TS in IP

Several methods are available for the encapsulation of an MPEG2 TS in IP datagrams. The most commonly used are:

- transport over RTP, as specified in ETSI EN 102 034;
- *ad hoc* mapping of TS packets to UDP datagrams, with one or more TS packets carried directly in a UDP datagram.

As all encryption is done within the transport stream, encapsulation is beyond the scope of this standard and the use of any particular encapsulation technique is no required. Encryption shall be applied within the transport stream and shall not be applied at the IP or RTP level.

Compliance with DVB-IPI is RECOMMENDED – direct UDP transport is likely to be added as an option to the DVB-IPI specifications (see ETSI EN 102 034) in the near future.

15.3 Delivery of traffic layer data in MPEG2 TS-based IP systems

When traffic layer data is transmitted over IP-based networks that use an MPEG2 transport stream, the traffic layer protection shall be applied within the MPEG2 transport stream, identical to the method specified in 14.2.

15.4 Delivery of key stream data in MPEG2 TS-based IP systems

In MPEG2 TS-based IP systems, the key stream data shall be delivered as specified in 14.3.

15.5 Delivery of rights management data in MPEG2 TS-based IP systems

In MPEG2 TS-based IP systems rights objects shall be delivered over an interactivity channel using the RO acquisition protocol (ROAP) as specified in OMA-ERP-DRM-V2_0. ROAP can be executed using different connectivity, such as HTTP, WAP, UPnP, OBEX over IrDA, Bluetooth or USB.

The interactivity channel used may be the Internet.

15.6 Delivery of registration data in MPEG2 TS-based IP systems

In MPEG2 TS-based IP systems, registration data shall be delivered over an interactivity channel using the RO acquisition protocol (ROAP) as specified in OMA-ERP-DRM-V2_0. ROAP may be executed using different connectivity technologies, such as HTTP, WAP, UPnP, OBEX over IrDA, Bluetooth or USB.

The interactivity channel used may be the Internet.

15.7 Signalling and service guides in MPEG2 TS-based IP systems

15.7.1 General

This subclause specifies how to perform signalling and service and content discovery in systems that use MPEG2 TS over IP to deliver services and content. It specifies a complete scheme for DVB-IPI (see ETSI EN 102 034) and sets out some requirements and recommendations for other systems that do not conform to DVB-IPI.

15.7.2 Signalling and the service guide in DVB-IPI systems

DVB-IPI (see ETSI 102 034) splits service discovery into three parts:

- bootstrapping via defined or discovered entry points;
- "SD&S" (service discovery and selection) – an XML-based description of IP-related data to locate a service; and
- PSI/SI data – carried in the transport stream in the usual way.

DVB-IPI (see ETSI EN 102 034) defines two SI profiles: TS full SI, which carries full, traditional SI information in the transport stream; and TS optional SI, which may carry only PAT and PMT tables with the rest of the information located in the SD&S data.

This subclause specifies the data that shall be carried in the SI and the SD&S, as an extension to the DVB-IPI specification; see ETSI EN 102 034.

15.7.2.1 Signalling of key stream messages

All DVB-IPI-based systems carry PAT and PMT. PAT and PMT shall conform to the requirements set out in 14.6.2.2, i.e. contain the necessary CA_descriptors to indicate key streams and encrypted component streams.

15.7.2.2 DVB-IPI SD&S signalling

15.7.2.2.1 Extensions to the broadcast discovery record

Where SDT is carried (for example, in a DVB-IPI "TS Full SI" system), it shall conform to the requirements set out in 14.6.2, i.e. contain the necessary CA_ID_descriptors. Any other discovery and purchase data above the SDT, PAT, PMT that may be carried in the stream shall be ignored and the data as specified in this clause used instead (that is PAT, PMT and SDT but not EIT).

Where SDT is not carried, SD&S shall carry the fields shown in Table 136, in addition to those specified in 5.2.6.2.1 and 5.2.6.2.2 of the DVB-IPI specification; see ETSI EN 102 034. The XML schema for these types is defined in 15.7.2.4. Additionally, it is RECOMMENDED that this information is carried in the SD&S in all cases.

Where both SD&S and SI information is available, precedence shall be as indicated in ETSI EN 102 034.

Table 136 – Additions to the broadcast discovery record

Broadcast discovery record	Attribute description	Mandatory/optional
Service(s)	For each service (in addition to already specified fields).	
CA ID identifier	Identifier of the broadcast protection system used. This shall be set to the value allocated to this standard by DVB, indicating that this standard is used.	M
SOC identifier	Identifier for the service operation centre responsible for the service.	O
ServiceBaseCID	The serviceBaseCID of this service; see 15.7.2.4.3.	O

If SOC identifier and serviceBaseCID are not carried in the broadcast discovery record, they shall be available via the service guide.

15.7.2.2.2 Extensions to the content-on-demand discovery record

The content-on-demand discovery record is extended to include the information shown in Table 137.

NOTE The use of this record will be deprecated in a forthcoming version of the DVB-IPI handbook, ETSI EN 102 034, and the broadband content guide discovery record should be used instead.

Table 137 – Additions to the content-on-demand discovery record

Content-on-demand discovery record	Attribute description	Mandatory/optional
Content-on-demand offerings(s)	For each offering (in addition to already specified fields).	
CA ID identifier	Identifies of the content-on-demand protection system available. This shall include the value allocated to this specification by DVB, indicating that this standard is used. Other protection systems may also be listed here.	M
SOC identifier	Identifier for the service operation centre responsible for the service.	O
ServiceBase CID	The serviceBaseCID of this service; see 15.7.2.4.3.	O

The device may use this field to determine whether it supports the content on demand protection system used. The relevant purchase information shall be included in the service guide, as specified in 15.7.2.3 or otherwise meet the requirements in 10.3.

The SOC identifier and serviceBaseCID identifier fields shall only be used where the values are consistent across the whole content-on-demand catalogue being advertised. More finely grained values can be specified in the service guide (see 15.7.2.3).

If SOC identifier and serviceBaseCID are not carried in the content-on-demand discovery record, they shall be available via the service guide.

15.7.2.2.3 Extensions to the package discovery record

No extensions to the package discovery record are required.

15.7.2.2.4 Extensions to the broadband content guide discovery record

No extensions to the broadband content guide discovery record are required.

15.7.2.3 Signalling of purchase data

All information relating to content items available for purchase should be signalled by the DVB-IPI broadband content guide, which uses TV anytime metadata. This standard provides a small number of extensions to TV anytime that allow full signalling of the required information. These extensions can be found in Clause A.19.

15.7.2.4 XML types

15.7.2.4.1 CA_ID_descriptor

This subclause specifies how the XML fragment is used to carry the CA_System_ID. This is equivalent to the CA_identifier_descriptor in ETSI EN 300 468.

```
<xs:complexType name="CA_ID_descriptor"/>
    <xs:sequence/>
        <xs:element name="CA_System_ID" type="anyURI"/>
    </xs:sequence>
</xs:complexType>
```

The value of CA_System_ID shall be of the format

```
urn:dvb:casystemid:
```

followed by the value assigned by DVB for this specification.

15.7.2.4.2 SOC identifier

The syntax of the SOC identifier is as specified in A.19.4.2.

15.7.2.4.3 ServiceBaseCID

The syntax of the serviceBaseCID is as specified in A.19.4.3.

15.7.3 Signalling and service guides in non-DVB-IPI systems

For systems based on MPEG2 TS over IP that do not conform to DVB-IPI, it is not possible to specify a complete scheme for the signalling and service/content discovery required for this standard. The following requirements are made:

- the requirements set out in 10.2, regarding signalling, shall be met;
- the requirements set out in 10.3, regarding service and content discovery, shall be met.

Furthermore, the following recommendations are made.

- Where signalling and/or service and content discovery is based on DVB-SI, it is RECOMMENDED that the mechanisms specified in 14.6.2 are followed as far as possible.
- Where the signalling used is the same or similar to DVB-IPI, it is RECOMMENDED that the mechanisms specified in this standard are followed as far as possible.
- Where the service and content discovery used is based on TV anytime, it is RECOMMENDED that the mechanisms specified in Clause A.19 are followed as far as possible.

15.8 Format of RI services over MPEG2 TS-based IP systems

MPEG2 TS-based IP systems do not employ a broadcast channel for the delivery of rights management and registration data. Therefore, RI services have not been defined for these systems.

15.9 Content-on-demand support

15.9.1 General

This standard inherently supports protection for content-on-demand services and can be used to build secure, flexible end-to-end systems.

Two kinds of protection are possible for content-on-demand services, and real world systems will probably use both.

- Content-on-demand items are delivered only to devices which are entitled to receive them (this is beyond the scope of this standard).
- Restrictions on viewing are enforced cryptographically and/or with the OMA DRM 2.0 rights expression language using this standard.

From a technical point of view, content on demand streams are no different to other streams, and require the same protection. The rights expression language provided by OMA DRM 2.0 is rich enough to implement most business models. Table 138 shows a possible sequence for the purchase and supply of a content-on-demand item.

Table 138 – Sequence of events for purchase and supply of a content-on-demand item

Step	Action by device	Action by COC	Action by SOC
1	User selects and confirms purchase of a CoD item from the service guide.		
2	Using the information in the service guide, device makes a ROAP request to the relevant COC.		
3		COC receives request and issues rights object Optionally – COC informs SOC of issue of rights object	
4	Device receives rights object Device makes request to SOC via, for example, RTSP to start CoD streaming.		
5			Optionally – SOC confirms whether rights object has been issued to this user SOC starts streaming CoD item.
6	Device receives CoD stream and is able to decrypt it using the rights object.		
7			Optionally – SOC informs COC when user has watched some proportion of the CoD item, for billing purposes.

The exact point that the user is charged for an item is beyond the scope of this standard. It could be done after step 3, step 5 or step 7.

The communication between the SOC and the COC is beyond the scope of this standard.

15.9.2 Content-on-demand trick play support

Some content-on-demand systems allows trick play of CoD items. This does not require any special consideration from this standard, but it is noted that SOCs are RECOMMENDED to ensure the following.

- Key stream messages remain synchronized with the encrypted traffic at all times.
- Where rights objects containing time constraints are issued, they should allow sufficient time for a user to watch the requested content, including some use of trick play features such as pause and rewind.

The enforcement of any restrictions on trick play, such as preventing a user from viewing a piece of content twice by restricting the use of a rewind feature, can be enforced using the OMA DRM 2.0 rights expression language to impose a time constraint, or simply at the CoD server by restricting use of the trick play functionality.

15.10 Use of server-side purchase interfaces

15.10.1 General

Some IPTV systems may use some server-side purchase interface (for example, a web shop), rather then advertising purchase data via the BCG. The operation of such an interface is beyond the scope of this standard. However, 15.10.2 and 15.10.3 explain how such a system is compatible with devices implementing this standard.

It should be noted that all systems shall meet the requirements for signalling in 10.2.

15.10.2 Example showing registration via a web interface

The sequence below illustrates how registration could take place via a web interface.

- The user navigates to the relevant website, where a rights issuer (or other party on behalf of a rights issuer) offers a web-based registration facility.
- The user and the website interact to provide the necessary information, the registration is confirmed and the rights issuer informed.
- The rights issuer sends a ROAP Trigger (see OMA-ERP-DRM-V2_0) to the device requesting the device to initiate the registration process.
- The device (perhaps after confirming with the user) initiates the 4-pass registration protocol; see OMA-ERP-DRM-V2_0.
- The protocol completes and registration is complete.

15.10.3 Example showing purchase via a web interface

The sequence below illustrates how a purchase could be made via a web interface.

- The user logs into the website in order to identify himself and his device.
- The user browses the web site and selects content to purchase.
- The user and the website interact in some way to confirm the purchase, after which the shop instructs a rights issuer to issue rights.
- The rights issuer either
  - issues a ROAP trigger to the device requesting the device to start the 2-Pass ROAP (see OMA-ERP-DRM-V2_0) to acquire the rights object; or
  - uses the 1-Pass ROAP to deliver the rights object direct to the device.

How the device subsequently receives or acquires the content is beyond the scope of this standard.

16 Protection of non-MPEG2 TS-based IP systems

16.1 General

This clause specifies the use of specific protection technologies from Clauses 6, 7, 8, 9, 10 and 11 for IP-based broadcast systems that do not employ an MPEG2 transport stream for the encapsulation of the content (see Clause B.11 for general references to IP-based broadcast systems). Clauses 4, 5 and 12 are general and apply to all systems.

16.2 Delivery of traffic layer data in non-MPEG2 TS-based IP systems

Table 139 shows how the traffic layer options are used when traffic layer data is transmitted over IP-based networks that do not use an MPEG2 transport stream.

Table 139 – Traffic layer options for transmission over non-MPEG2 TS based IP networks

Traffic layer	Transmission details
IPsec	As specified in IETF RFC 4301
ISMACryp	As specified in ISMACryp 2.0
SRTP	As specified in IETF RFC 3711 and IETF RFC 4771
MPEG2 TS	Not applicable

16.3 Delivery of key stream data in non-MPEG2 TS-based IP systems

In IP systems not based on MPEG2 TS, each KSM shall be encapsulated in exactly 1 UDP packet. The KSM data shall be carried in a distinct IP flow, separated from other data.

In order to keep access times low for devices that start accessing a multicast service, a KSM shall be transmitted periodically when multicast.

16.4 Delivery of rights management data in non-MPEG2 TS-based IP systems

The way non-MPEG2 TS-based IP systems obtain rights objects is identical to the method specified in 15.5.

16.5 Delivery of registration data in non-MPEG2 TS-based IP systems

The way in which non-MPEG2 TS-based IP systems registration data is delivered is identical to the method specified in 15.6.

16.6 Signalling and service guides in non-MPEG2 TS-based IP systems

The difference between MPEG2 TS and non-MPEG2 TS-based IP systems is that the content itself is not encapsulated in an MPEG2 transport stream, but instead carried directly over another protocol offering facilities for timing and synchronization; for example, RTP, see IETF RFC 3550.

There are currently no DVB-IPI standards for non-TS-based systems. In the absence of such standards, this standard mandates only that signalling and content discovery within such systems shall meet the requirements set out in 10.3. However, it is anticipated that the signalling and content discovery mechanisms for MPEG2 TS-based IP systems can mostly be reused for non-MPEG2 TS-based IP systems.

16.7 Format of RI services over non-MPEG2 TS-based IP systems

Non-MPEG2 TS-based IP systems do not employ a broadcast channel for the delivery of rights management and registration data. Therefore, RI services have not been defined for such systems.

16.8 Content-on-demand support

It is anticipated that the considerations specified in 15.9 will also apply to non-MPEG2 TS-based IP systems.

Annex A
(normative)

Supporting specifications

A.1 Security considerations

A.1.1 Handling weak keys

The responsible components in the head-end architecture shall not use weak keys for messages in the IP Datacast network. At the time of writing, there are no specified weak keys for use in AES. This does not imply that weak keys do not exist. If, at some point, a set of weak keys for AES is identified, the use of these weak keys shall be rejected in the head-end architecture followed by a request for replacement key.

A.1.2 Handling OCSP grace period

If a device without a return channel inspects a certificate, because the user wants to consume certain content for which he has acquired the RO, and the device finds out that the OCSP response of the certificate chain has expired, then the device shall still be allowed to use it for a short period of time during which the user has time to set the process in motion through which the device will receive a new OCSP response. This means that the user can enjoy the content he was entitled to consume straight away, at the expense of allowing the use of possibly compromised certificates for a short grace period.

A device in broadcast-only mode shall implement the grace period mechanism.

a) The device shall check periodically a particular or all RI contexts for expiration.

b) If a RI context is expired, the device displays an OCSP response expiry reminder for the associated RI context. The reminder notifies the user that the user needs to get a new OCSP response. This should be done of course in terms that a user can understand like "call this number with this message please".

c) Until this OCSPresponse expiry reminder is invoked, the device will be rendered inoperable, but only in relation with the associated RI (context) as specified below.

   1) Accessing a service guide for purchase is still allowed.

   2) The device shall be rendered inoperable for any purchase protocol or playback of future content. The device may use stored BCROs to play old content for which the device obtained ROs, but shall not use these BCROs for new content received after the re-registration request until the device received a fresh OCSP response or is re-registered with the RI.

d) A device shall be allowed to use an expired OCSP response for a pre-defined grace period. The grace period shall not be more than the OCSP response's lifetime (the difference between the nextUpdate and thisUpdate fields in the OCSP response), and shall not exceed 48 h. During the grace period, the device can use the expired OCSP response.

   1) The grace period is for a one-time use only.

   2) The terminal shall support secure DRM time.

   3) Rules in ROs shall have precedence over the OCSP response grace period usage.

e) If the secure timer (i.e. grace period) expires and a fresh OCSP response has not been received, the device will be rendered inoperable, but only in relation with the associated RI (context) as specified below.

   1) Accessing a service guide for purchase is still allowed.

2) The device shall be rendered inoperable for any purchase protocol or playback of future content. The device may use stored BCROs to play old content for which the device obtained ROs, but shall not use these BCROs for new content received after the re-registration request until the device received a fresh certificate chain or is re-registered with the RI.

A device in broadcast mode may implement a mechanism to schedule the certificate chain updates automatically.

f) An update (powerup/powerdown) timeslot is programmed in which the RI will transmit the certificate chain. The timeslot may be obtained from the service guide. The device should parse the received service guide data to find a time at which it can receive a certificate chain update. It may be the case that certificate chain updates are broadcast continuously. See 13.7.7 and 14.9 for more details.

g) Upon power down before update, the device may display a warning message that the device needs to update its device chain. An example might look like: "do not power off device. Device will perform update during xx:yy h".

The device will be powered up and down in timeslot xx:yy h to pick up the message to update the RI certificate chain (notably the OCSP response).

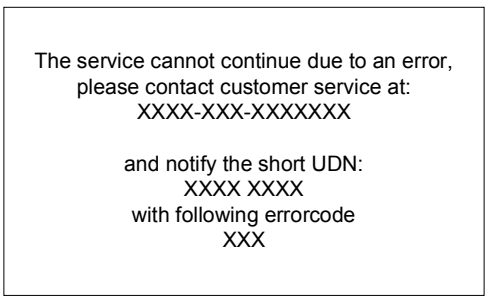
A.2 Status and error message handling

This clause specifies the status and error values that shall be used in the 1-pass protocols for broadcast devices.

The status field is a binary value. Upon receipt of a message for which status is not "success", the default behaviour, unless explicitly stated otherwise below, is that both the RI and the device shall immediately close the connection and terminate the protocol. RI systems and devices are required to delete any session-identifiers, nonces, keys, and/or secrets associated with a failed run of the protocol.

When possible, the device should present an appropriate error message to the user. An example of such a message is shown in Figure A.1.

It is the sequence of the defined values that is specified. The use of dashes as the delimiter is shown with an example placement to be consistent with the examples used elsewhere in this standard. The text portion of this screen is shown as an example only; there is no implied requirement to duplicate the exact wording or formatting shown. The numeric fields shall be included as defined above. The short UDN will only be displayed after the first registration, when that data is available for display.



An example dialogue showing an error.

NOTE The error codes should be displayed as a three digit decimal number. See Table A.1 for an overview of possible error codes.

Figure A.1 – Sample notification display

Table A.1 – Status/error codes

Status/Error	Value$_{(h)}$	Comment
Success	0x00	
UnknownError	0x01	
Abort	0x02	
NotSupported	0x03	
AccessDenied	0x04	
NotFound	0x05	
MalformedRequest	0x06	
UnknownRequest	0x07	
UnsupportedVersion	0x08	
NoCertificateChain	0x09	
SignatureError	0x0A	
DeviceTimeError	0x0B	
NotRegistered	0x0C	
InvalidDomain	0x0D	
DomainFull	0x0E	
TokenConsumptionMessageError	0x0F	
NoTokenConsumptionMessage	0x10	
ForceInteractiveChannel	0x11	
ForceOobChannel	0x12	
Reserved for future use	0x13-0xFF	

UnknownError indicates an internal RI system error.

Abort indicates that the RI rejected the device's request for unspecified reasons.

NotSupported indicates the device made a request for a feature currently not supported by the RI.

AccessDenied indicates that the device is not authorized to contact this RI.

NotFound indicates that the requested object was not found.

MalformedRequest indicates that the RI failed to parse the device's request.

UnknownRequest indicates that the RI did not recognize the request type.

UnsupportedVersion indicates that the device used a ROAP protocol version not supported by the RI.

NoCertificateChain indicates that the RI could not verify the signature on a device request due to not having access to the device's certificate chain.

SignatureError indicates that the RI could not verify the device's signature.

DeviceTimeError indicates that rights issuer request a device to set the device DRM time with a new value and report the time drift to the rights issuer.

NotRegistered indicates that the device tried to contact an RI with which it has not completed a valid registration. The RI should include the RI ID attribute in the response message in which this error code is sent. The device should perform the off-line device registration protocol until a retry limit but shall have user consent to start.

InvalidDomain indicates that the request was invalid due to an unrecognized domain identifier.

DomainFull indicates that no more devices are allowed to join the domain.

TokenConsumptionMessageError indicates that the RI did receive a token consumption message, but that it was erroneous and that the device should redo the last token consumption message.

NoTokenConsumptionMessage indicates that the RI did not receive a token consumption message yet, but was expecting one, because the present date/time is later than the last latest_token_consumption_time sent to the device in a token delivery response message.

ForceInteractiveChannel indicates that the RI forces a mixed-mode device to use its interactive channel exclusively and not its OOB channel.

ForceOobChannel indicates that the RI forces a mixed-mode device to use its OOB channel exclusively and not its interactive channel.

A.3 Time and date conventions

A.3.1 Specification of the mjdutc format

The mjdutc format is a 40-bit field that represents date and time.

The first (left) 16 bits contain the 16 least significant bits of MJD from A.3.2.

If the first (left) 16 bits of the mjdutc field are less than 15079, these 16 bits represent a value for *MJD* of 65536 + the value of the first (left) bits of the mjdutc field.

NOTE The first (left) 16 bits of the mjdutc field represent the inclusive dates 1900 March 1 to 2081 April 25.

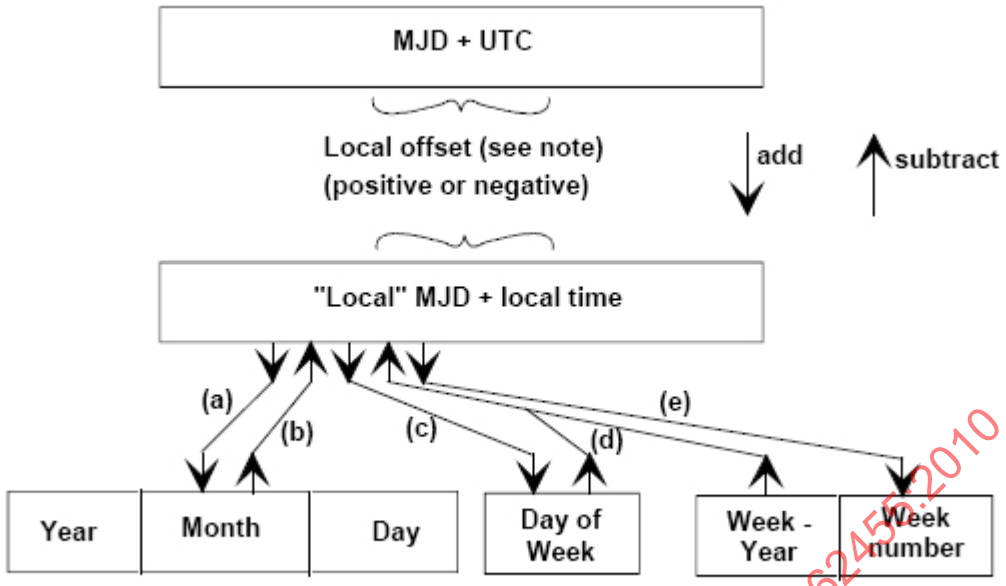
The last (right) 24 bits of the mjdutc field is the UTC field, which represents time. The UTC field is coded as 6 digits in 4-bit binary coded decimal (BCD).

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

A.3.2 Conversion between time and date conventions

NOTE 1 This text originates from ETSI EN 300 468 V1.6.1 but has been slightly modified. The version in ETSI EN 300 468 V1.6.1 has a byte alignment problem caused by the time offset polarity field. The version in this text maintains byte alignment everywhere.

The types of conversion, which may be required, are summarized in Figure A.2.



NOTE: Offsets are positive for Longitudes East of Greenwich and negative for Longitudes West of Greenwich.

Figure A.2 – Conversion routes between modified julian date (MJD) and coordinated universal time (UTC)

The conversion between MJD + UTC and the "local" MJD + local time is simply a matter of adding or subtracting the local offset. This process may, of course, involve a "carry" or "borrow" from the UTC affecting the MJD. The other five conversion routes shown on the diagram are detailed in the formulas below.

Symbols used:

D	Day of month from 1 to 31
int	Integer part, ignoring remainder
K, L, M', W, Y'	Intermediate variables
M	Month from January (= 1) to December (= 12)
MJD	Modified Julian Date
WN	Week number according to ISO 8601
mod 7	Remainder (0-6) after dividing integer by 7
UTC	Universal Time, Co-ordinated
WD	Day of week from Monday (= 1) to Sunday (= 7)
WY	"Week number" Year from 1900
×	Multiplication
Y	Year from 1900 (for example, for 2003, *Y* = 103)

a) To find Y, M, D from MJD

Y' = int [(MJD – 15 078,2) / 365,25]

M' = int { [MJD – 14 956,1 – int (Y' × 365,25)] / 30,600 1 }

D = MJD – 14 956 – int (Y' × 365,25) – int (M' × 30,600 1)

If M' = 14 or M' = 15, then K = 1; else K = 0

Y = Y' + K

M = M' – 1 – K × 12

b) To find MJD from Y, M, D

If M = 1 or M = 2, then L = 1; else L = 0

MJD = 14 956 + D + int [(Y – L) × 365,25] + int [(M + 1 + L × 12) × 30,600 1]

c) To find WD from MJD

WD = [(MJD + 2) mod 7] + 1

d) To find MJD from WY, WN, WD

MJD = 15 012 + WD + 7 × { WN + int [(WY × 1 461 / 28) + 0,41] }

e) To find WY, WN from MJD

W = int [(MJD / 7) – 2 144,64]

WY = int [(W × 28 / 1 461) – 0,007 9]

WN = W – int [(WY × 1 461 / 28) + 0,41]

EXAMPLE	MJD = 45 218	W = 4 315
	Y = (19)82	WY = (19)82
	M = 9 (September)	N = 36
	D = 6	WD = 1 (Monday)

NOTE 2 These formulas are applicable between the inclusive dates 1900 March 1 to 2100 February 28.

A.3.3 Local time offset

This 16-bit field contains the current offset time from UTC in the range between –12 h and +13 h at the area that is indicated by the combination of country_code and country_region_id in advance. These 16 bits are coded as 4 digits in 4-bit BCD in the order hour tens, hour, minute tens, and minutes.

The positive or negative offset from the UTC is indicated with the 1-bit local_time_offset_polarity. If this bit is set to '0' the polarity is positive and the local time is advanced to UTC (usually east direction from Greenwich). If this bit is set to '1' the polarity is negative and the local time is behind UTC. The local_time_offset_polarity is represented by the first bit of the first nibble representing the hour tens field. The first nibble of the local_time_offset is therefore encoded as specified in Table A.2.

Table A.2 – Local time offset coding

Local_time_offset_polarity	Offset hour tens	First nibble
0 (i.e. "+")	0	0000
0 (i.e. "+")	1	0001
1 (i.e. "-")	0	1000
1 (i.e. "-")	1	1001

A.4 Conversion of OMA DRM 2.0 content identifiers to binary content identifiers

The CID in OMA DRM 2.0 has the form of a URI. For a broadcast rights object as defined in 8.4.2, this text base id has to be transformed into a binary id. The ID is split into a base_BCI and an extension_BCI.

Definition:

Field	Description/Value	Type
socID	Service operator centre ID as signalled in service guide	String
content_id	Textual id as used in OMA DRM 2.0 and signalled in the service guide	String
content_type	"#P" for programmes and "#S" for services	String
cid-url	"cid:" || socID || content_type || content_id || "@"	String

The 64-bit binary content id BCI is then given by

base_BCI = HMAC-SHA1-64(cid-url)

The extension_BCI is a 32-bit id (see programme_CID_extension and service_CID_extension in the KSM).

The BCI is given by

$$BCI = (base\_BCI << 32) \,|\, extension\_BCI$$

A.5 Conversion of OMA X509SPKIHash values to binary values

Values codes as roap:X509SPKIHash in OMA DRM 2.0 (see OMA-TS-DRM-DRM-V2_0) are SHA1-160 hashes. In OMA DRM 2.0, these values are presented as base64 encodings in a XML <hash> element as shown by the example below.

```
<keyIdentifier xsi:type="roap:X509SPKIHash">
    <hash>aXENc+Um/9/NvmYKiHDLaErKofk=</hash>
</keyIdentifier>
```

If such a hash is carried in a binary object, for example, the BCRO or a KSM, the 160-bit hash itself without the base64 encoding will be used.

A.6 Limits of the surplus_block()

A.6.1 General

The following examples show two possible cases: one keyset for standards addressing and one keyset with additional domain addressing.

A.6.2 Standard keyset

For standard addressing is keyset_block filled with

- 1 UGK, 9 SGK, 1UDK, 1 UDF, 1 RIAK, 1 UDF, see Table A.3 and Table A.4.

The maximum subscriber group size is 512, which is supported by 9 SGK.

Table A.3 – Standard keyset with RSA block size 1024

Value	Variable	Key size	Key data	Key size	Key data	Key size	Key data
1024	RSA size						
1	UDF	40	40	40	40	40	40
1	UGK	128	128	128	128	128	128
9	SGK	128	1 152	128	1 152	128	1 152
1	UDK	128	128	128	128	128	128
1	RIAK	128	128	128	128	128	128
0	BDK	128	0	128	0	128	0
0	SBDF	48	0	48	0	48	0
0	LBDF	840	0	840	0	840	0
0	TDK	128	0	128	0	128	0
13	TLF overhead	7	181	7	181	7	181
	keyset_block		**1 757**		**1 757**		**1 757**
1	SK	128	128	192	192	256	256
0	PKCS overhead		0		0		0
	sessionkey_block()	1 024		1 024		1 024	
	Room in sessionkey_block() to use for keyset_block		896		832		768
	(remainder of keyset_block in) surplus_block()		**861**		**925**		**989**

Other block sizes with keyset as depicted above produce the following results.

Table A.4 – Standard keyset with other RSA block sizes

Block size	SK size	Keyset_block	Room in sessionkey_block()	Surplus block
2 048	128	1 757	1 920	No
2 048	192	1 757	1 856	No
2 048	256	1 757	1 792	No
4 096	128	1 757	3 968	No
4 096	192	1 757	3 904	No
4 096	256	1 757	3 840	No

A.6.3 Extended keyset

An extended keyset includes keys for standard addressing plus domain addressing. The keyset_block is filled with

- 1 UGK, 9 SGK, 1UDK, 1 UDF, 1 RIAK, 1 UDF, 1 BDK, 1 SBDF, 1 LBDF (maximum size), 1 TDK (see Table A.5 and Table A.6)

The maximum subscriber group size is 512, which his is supported by 9 SGK.

Table A.5 – Extended keyset with RSA block size 1024

Value	Variable	Key size	Key data	Key size	Key data	Key size	Key data
1024	RSA size						
1	UDF	40	40	40	40	40	40
1	UGK	128	128	128	128	128	128
9	SGK	128	1 152	128	1 152	128	1 152
1	UDK	128	128	128	128	128	128
1	RIAK	128	128	128	128	128	128
1	BDK	128	128	128	128	128	128
1	SBDF	48	48	48	48	48	48
1	LBDF	840	840	840	840	840	840
1	TDK	128	128	128	128	128	128
17	TLF overhead	7	219	7	219	7	219
	keyset_block		**2 939**		**2 939**		**2 939**
1	SK	128	128	192	192	256	256
0	PKCS overhead		0		0		0
	sessionkey_block()	1 024		1 024		1 024	
	Room in sessionkey_block() to use for keyset_block		896		832		768
	(remainder of keyset_block in) surplus_block()		**2 043**		**2 107**		**2 171**

Table A.6 – Extended keyset with other RSA block sizes

Block size	SK size	Keyset_block	Room in sessionkey_block()	Surplus block
2 048	128	1 924	1 920	1 019
2 048	192	1 924	1 856	1 083
2 048	256	1 924	1 792	1 147
4 096	128	1 924	3 968	No
4 096	192	1 924	3 904	No
4 096	256	1 924	3 840	No
NOTE Not yet included is the PKCS overhead in the sessionkey_block(), so surplus_block() will be a little larger.				

A.7 Function F_{ZMB}

A.7.1 Definitions

The function that shall be used for the ZMB system is defined as follows.

a) **node numbering calculation**. The keys in the zero message broadcast encryption key tree are numbered in a breadth-first fashion. The root key has number 0 (NK_0). For any parent key NKi, the left child key is NK2i+1 and the right child key is NK2i+2, as shown in Figure A.3.

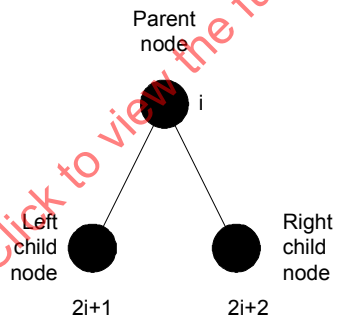


Figure A.3 – Node numbering

b) **node/leaf key derivation function**. Given a parent key NKi, the derived child keys NK2i+1 and NK2i+2 are calculated by encrypting a known constant using AES as shown in Figure A.4.

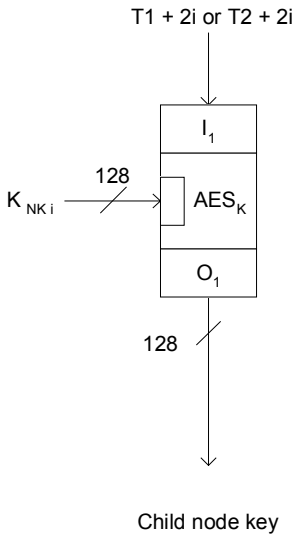


Figure A.4 – AES for key derivation

Explanation of Figure A.4.

- Step 1: Initialize the system with following values.
  - Let T1 = 0x01010101010101010101010101010101
  - Let T2 = 0x02020202020202020202020202020202
  - Load $K_{NK\ i}$ (bslbf), where $K_{NK\ i}$ = SGK (in case of device) or $K_{NK\ i}$ = Root Key NK_0 (in the case of RI)
- Step 2: Calculate the left child node and the right child node from an input key SGK. Or, given a parent key NKi (i.e. root key or SGK), the derived child keys NK2i+1 (i.e. left child node) and NK2i+2 (i.e. right child node) are specified by following functions.

  NK2i+1 = AES_ENC_128{NKi}(2i+T1)

  NK2i+2 = AES_ENC_128{NKi}(2i+T2)

  AES shall be compliant with FIPS PUB 197:2001.

c) **Computation of all keys.** The rights issuer computes all keys in the tree by recursively applying the relations given by (b) starting from the root key NK0 that is known only to the rights issuer.

d) **Leaf number calculation**. In a subscriber group for n devices, these n devices are associated with the leaf keys NKn-1 up to and including NK2n-2. A device with position p (numbered from 0 to n-1) in the group is associated with key NKp+n-1.

e) **Determining the (SGK) keyset of a device.** Define the functions sibling(i) and parent(i) as follows.

```
sibling(i) = { i-1 if i is even,
               i+1 if i is odd }.

parent(i) = { i/2 - 1 if i is even
            (i-1)/2 if i is odd }
```

Then the following algorithm defines the key set to be given to a device with position p in the group.

```
i:= p+n-1
KeySet = {}
while (i > 0) {
    i:= sibling(i)
    KeySet = KeySet union { NKi }
    i:= parent(i)
} //end while
```

f) **Determining the exclusion keys.** Given an access mask, it is possible to determine the key numbers of the keys used as exclusion keys. Each device given its own keyset can determine these exclusion keys, except if the key associated with its own position is used as an exclusion key. To effectively disallow devices to access a certain asset, the rights issuer derives an IEK by concatenating the device revocation keys and using this concatenation as key for computing a MAC over the BCI as retrieved from the rights object:

$$IEK = HMAC - SHA1\_128\{NK_{ex-1} || NK_{ex-2} || \ldots || NK_{ex-n}\}(BCI)$$

g) **IEK calculation.** The notation HMAC_SHA1(k, s) is used to denote the computation of HMAC (see IETF RFC 2104) keyed by the key 'k' over the string 's' with SHA1 (see FIPS PUB 180-2:2002) as the hash function. HMAC_SHA1_128(k, s) is used to denote the 128 most significant bits of HMAC_SHA1(k, s) output.

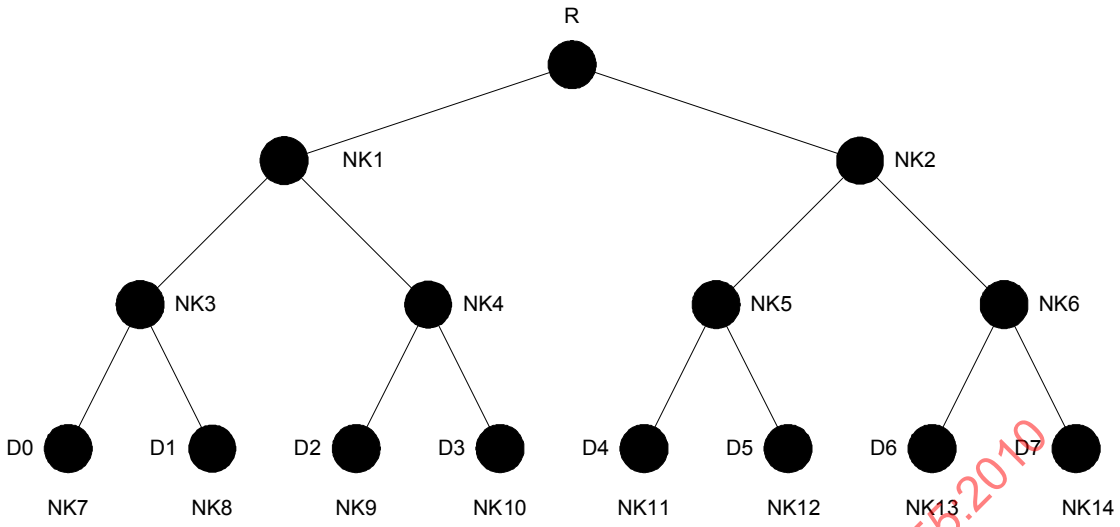
NOTE IETF RFC 2104 specifies that if the key is longer than the output block length of SHA1 then the key is first hashed to the SHA1 block length. This is expected to be the case frequently in this algorithm. A device that has verified that its own key is not part of the exclusion key set, can calculate the IEK using the following algorithm:

```
function CalculateEncryptionKey
{
    IEK = 0
    b:= 0
    Temp:= ""
    while (b < n) {
        if (bit b in access_mask equals 0) {
            Temp:= Temp || CalculateNodeKey(b + n - 1)
        } //end if
    } //end while
    IEK:= HMAC_SHA1_128(Temp, Salt)
    return IEK
}

function CalculateNodeKey(i)
{
    if (NKi is in KeySet) {
        return NKi from KeySet
    }
    else {
        parentkey = CalculateNodeKey( parent(i) )
        return AES{ parentkey }( i )
    }
}
```

A.7.2 Example tree and keyset

Figure A.5 shows a sample tree with keys for a device at position 7 (i.e. device D0), as defined in subsection 'e' of A.7.1



EXAMPLE 1 Keyset for D0 = {NK2, NK4, NK8}

EXAMPLE 2 Keyset for D4 = {NK1, NK6, NK12}

EXAMPLE 3 To effectively disallow devices D1, D5 and D7 to access a certain asset, the rights issuer derives an IEK by concatenating the device revocation keys (NK_8, NK_{12} and NK_{14}) and using this concatenation as a key for computing an HMAC-SHA1-128 over the BCI as retrieved from the rights object:

$$IEK = HMAC - SHA1\_128\{NK_8 \,||\, NK_{12} \,||\, NK_{14}\}(BCI)$$

Figure A.5 – Sample tree with correct node and device numbering

A.8 Authentication

A.8.1 General

For a quick overview of the authentication "hierarchy" of the system, see 5.4.8.

A.8.2 Authentication for IPsec

IPsec may be used with authentication. In case of authentication with IPsec, the authentication data shall carry the TAS. The authentication mechanism shall create the TAK from the TAS.

To obtain the encrypted traffic key material from the KSM the encrypted traffic key material shall be decrypted with the SEK or PEK:

$$TAS = D\{SEK\}(traffic\_key\_material)$$

or

$$TAS = D\{PEK\}(traffic\_key\_material)$$

The authentication key shall be generated from the authentication seed:

$$TAK = f_{auth}\{TAS\}(CONSTANT\_TAK)$$

where

$$CONSTANT\_TAK = 0x040404040404040404040404040404 \text{ (120 bit)}$$

See A.8.5 for details on f-auth.

The TAK shall be used in the MAC generation/verification of the IPsec data. See IETF RFC 2406 for details.

A.8.3 Authentication for KSMs

A.8.3.1 General

A key stream message can contain two MAC fields, the programme MAC field and the service MAC field. If only one MAC field was used, the authentication key could only be renewed when both SEK and PEK change at the same time. Having two MAC fields and two authentication keys makes it possible to authenticate the message and check for its integrity while only having one key set. The service authentication key (SAK) and the programme authentication key (PAK) shall be derived from the service authentication seed and the programme authentication seed respectively, which are transmitted together with the encryption keys in the ROs. How this is carried in the BCRO and ICRO is defined in subsequent subclauses. A service RO will contain SEAKs and a programme RO will contain PEAKs.

To obtain the PAS or SAS from the BCRO, the encrypted SEAK/PEAK, see 8.4.3, shall be decrypted with the IEK:

$$SAS = LSB_{128}(D\{IEK\}(encrypted\_service\_encryption\_authentication\_key))$$

$$PAS = LSB_{128}(D\{IEK\}(encrypted\_program\_encryption\_authentication\_key))$$

The authentication key shall be generated from the authentication seed:

$$SAK = f_{auth}\{SAS\}(CONSTANT\_SAK)$$

$$PAK = f_{auth}\{PAS\}(CONSTANT\_PAK)$$

where

CONSTANT_SAK = 0x02020202020202020202020202020202 (120 bit)

CONSTANT_PAK = 0x01010101010101010101010101010101 (120 bit)

See A.8.5 for details on f-auth.

The SAK or PAK shall be used in the MAC generation/verification of the KSM. The algorithm used to calculate the MAC field shall be HMAC-SHA1-96 according to FIPS PUB 198:2002 and IETF RFC 2104, using authentication keys of 160 bit in both cases.

A.8.3.2 Transport of SEAK and PEAK in OMA DRM 2.0 rights objects

The encryption keys and authentication keys (SEAK and PEAK), encrypted with a key transport algorithm as specified in OMA-ERP-DRM-V2_0 (the default being AES_wrap (see NIST:2001) which is used in the examples below), shall be transported in an ICRO as separate ds:KeyInfo elements in the <asset> fragment of the rights object. Hence, the <asset> fragment of a service ICRO contains the following (see OMA-TS-DRM-REL-V2_0 for a detailed specification of the XML elements).

```
<o-ex:asset o-ex:id="asset_ID">

[…]

  <ds:KeyInfo>
    <xenc:EncryptedKey>
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
      <ds:KeyInfo>
          <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>encrypted_service_encryption_key</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedKey>
  </ds:KeyInfo>
  <ds:KeyInfo Id="service_authentication_seed_id" >
    <xenc:EncryptedKey>
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
      <ds:KeyInfo>
          <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>encrypted_service_authentication_seed</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedKey>
  </ds:KeyInfo>
</o-ex:asset>
```

encrypted_service_encryption_key =

$$E\{REK\}(SEK) = AES\_wrap\{REK\}(SEK)$$

encrypted_service_authentication_seed =

$$E\{REK\}(SAS) = AES\_wrap\{REK\}(SAS)$$

where SEK and SAS are both an AES key of 128 bits and service_authentication_seed_id is a unique identification of the authentication seed KeyInfo element within the ICRO, constructed as follows.

```
service_authentication_seed_id = asset_ID || "_authSeed"
```

Similarly, the <asset> element of a programme ICRO contains

```
<o-ex:asset o-ex:id="asset_ID">

[…]

  <ds:KeyInfo>
    <xenc:EncryptedKey>
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
      <ds:KeyInfo>
          <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
      </ds:KeyInfo
      <xenc:CipherData>
        <xenc:CipherValue>encrypted_programme_encryption_key</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedKey>
  </ds:KeyInfo>
  <ds:KeyInfo Id="programme_authentication_seed_id" >
    <xenc:EncryptedKey>
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
      <ds:KeyInfo>
          <ds:RetrievalMethod URI="#K_MAC_and_K_REK"/>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>encrypted_programme_authentication_seed</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedKey>
  </ds:KeyInfo>
</o-ex:asset>
```

encrypted_programme_encryption_key =

E{REK}(PEK) = AES_wrap{REK}(PEK)

encrypted_programme_authentication_seed =
E{REK}(PAS) = AES_wrap{REK}(PAS)

where PEK and PAS are both an AES key of 128 bits and programme_authentication_seed_id is a unique identification of the authentication seed KeyInfo element within the ICRO, constructed as follows.

```
programme_authentication_seed_id = asset_ID || "_authSeed"
```

A.8.3.3 Transport of SEAK and PEAK in BCROs

The encryption keys and authentication keys (SEAK and PEAK) shall be transported in a BCRO by concatenating the encryption key and the authentication seed and then protecting the resulting field with AES CBC.

encrypted_service_encryption_authentication_key =
E{IEK}(SEAK) = AES_CBC{IEK}(SEK << 128) || SAS)

where SEK and SAS are both an AES key of 128 bits.

and encrypted_programme_encryption_authentication_key =
E{IEK}(PEAK) = AES_CBC{IEK}(PEK << 128) || PAS)

where PEK and PAS are both an AES key of 128 bits.

A.8.4 Authentication of BCROs

BCROs contain one MAC field that is used to authenticate the message and to protect the integrity of the message.

The authentication key shall be generated from the RIAK:

$$BAK = f_{auth}\{RIAK\}(CONSTANT\_BCRO)$$

where

CONSTANT_BCRO = 0x030303030303030303030303030303 (120 bit)

To obtain the RIAK the device needs to have been equipped with a valid keyset. See 9.3.2 for details.

See A.8.5 for details on f-auth.

The BAK shall be used in the MAC generation/verification of the BCRO. The algorithm used to calculate the MAC field is HMAC-SHA1-96 according to FIPS PUB 198:2002 and IETF RFC 2104, using an authentication key of 160 bits.

A.8.5 General authentication mechanism

The function F-auth shall consist of the following steps.

a) Denote by PRF{key}(text) as the AES-XCBC-MAC-PRF with output block size 128 bits as defined by IPsec WG in IETF.
   - See IETF RFC 3566 for the AES-XCBC-MAC-PRF based key generation function.
   - See IETF RFC 3664 for the requirement NOT to truncate the generated key material.

b) Apply the generated input key according to ideas of IKEv2 to generate authentication key. Define a key generator function f-kg{key}(constant). Keying material will always be derived

as the output of the negotiated PRF algorithm. PRF^+ is the function that outputs a pseudo-random stream of n blocks based on the inputs to a PRF as follows:

$$T1 = AES\_XCBC\_MAC\_PRF\{AS\}(CONSTANT \,||\, 0x01)$$

$$T2 = AES\_XCBC\_MAC\_PRF\{AS\}(T1 \,||\, CONSTANT \,||\, 0x02)$$

....

$$Tn = AES\_XCBC\_MAC\_PRF\{AS\}(T1 \,||\, CONSTANT \,||\, n)$$

where AS is the appropriate authentication seed (be it TAS, PAS, SAS or RIAK) and CONSTANT is the appropriate constant as specified in preceding subclauses or in the next one. The amount of blocks to derive is defined by the amount of key material needed, i.e. n is the amount of needed key bits divided by 128 and rounded up.

This means that if 160 bits were needed then $PRF^+()$ would be computed as:

$$T1 \,||\, T2 = PRF^+\{K\}(S)$$

c) The 160-bit authentication key is taken from the generated key material as follows:

$$AK = MSB_{160}(T1 \,||\, T2)$$

The generated authentication key shall be applied as specified in preceding subclauses.

A.8.6 Authentication of token delivery response messages

Token delivery response messages contain one MAC field that shall be used to authenticate the message and to protect the integrity of the message.

The authentication key shall be generated from the RIAK:

$$TDRMAK = f_{auth}\{RIAK\}(CONSTANT\_TDRM)$$

where

$$CONSTANT\_TDRM = 0x05050505050505050505050505050505 \quad (120 \text{ bit})$$

To obtain the RIAK the device needs have been equipped with a valid keyset. See 9.3.2 for details.

See A.8.5 for details on f-auth.

The TDRMAK shall be used in the MAC generation/verification of the token delivery response message. The algorithm used to calculate the MAC field shall be HMAC-SHA1-96 according to FIPS PUB 198:2002 and IETF RFC 2104, using an authentication key of 160 bit.

A.8.7 Computation of the TAA_report_code field in the TAA report

Devices shall compute the TAA_report_code field of the TAA report (see 9.3.2.3.8) in the way specified in this subclause.

During registration, the device may receive a TAA_descriptor (see 9.3.2.7.2 and A.11.7). When the device receives a present_TAA_report() message (see 9.3.2.7.7), the TAA report shall be computed immediately and shall be shown to the user without the user asking for it (for example, through the user interface of the device). The device shall compute the TAA

report and show it to the user whenever the user asks for it through the user interface of the device. The device shall have suitable user interface functionality for this. The way a TAA report should be shown to the user is specified in 9.3.2.3.8.

In the cases that it has to be computed, the TAA_report_code shall computed as follows.

The TAA_report_value from the TAA_descriptor padded at the right with zero-valued bits, such that a field with the block size of the TAA algorithm is obtained. This right-padded field is decrypted with the TAA algorithm and TAA parameter as indicated in the TAA descriptor. The leftmost 56 bits are of this decryption are taken as the TAA_report_code. The 17-digit decimal representation of these 56 bits, including any leading zeroes is used as the TAA_report_code in the TAA report. See also Figure A.6.

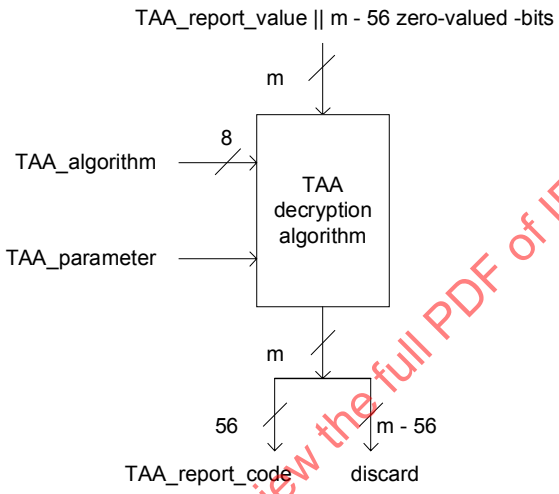


Figure A.6 – Computation of the TAA_report_code

A.9 Checksum algorithms

A.9.1 General

The likelihood of errors in human communication according to empirical research from Verhoef:1969 is expressed in Table A.7.

Table A.7 – Error likelihood in human communication

Nr	Error	Representation	Relative likelihood in %
1	Single substitution	a => b	60 to 95
2	Single adjacent transpositions	ab => ba	10 to 20
3	Twin errors	aa => bb	0,5 to 1,5
4	Jump transpositions (longer jumps are even rarer)	acb => bca	0,5 to 1,5
5	Phonetic errors (phonetic, because in some languages the two have similar pronunciation, for example, thirty and thirteen)	a0 => 1a where a={2,..,9}	0,5 to 1,5
6	Adding or omitting digits		10 to 20
a and b are different decimal digits, while c can be any decimal digit.			

The most common errors are therefore errors 1, 2 and 6. Error 6 is easily detected. The following subclauses define a method to detect other errors.

A.9.2 UDN checksum – Definition

The checksum on the UDN shall be calculated by F-UDN.

We use codes over $\mathbb{Z}_p$, the integers modulo p, where p = 11. That is to say, codewords are strings with entries from $\{0,1,....p-1\}$. We consider codes of length n defined by r parity equations: a string $\mathbf{c}=(c_1,c_2,....,c_n)$ with elements from $\mathbb{Z}_p$ is a codeword if, and only if, it satisfies the following condition

$$\sum_{j=1}^{n} a_j c_j \equiv 0 \,(\text{mod } p)$$

for each check equation, which is defined by $\mathbf{a}=(a_1,a_2,....,a_n)$.

We now describe a [20,17] code, that is defined over 20 symbols from $\mathbb{Z}_{11}$ using the three following check equations as specified in the matrix H3 below:

Take n = 17, r = 3 and p = 11. We consider the code defined by the r = 3 following check equations.

$$1 \times c_1 + 0 \times c_2 + 1 \times c_3 + ... + 1 \times c_{18} = 0 \text{ (modulo 11)}$$
$$0 \times c_1 + 1 \times c_2 + 0 \times c_3 + ... + 1 \times c_{19} = 0 \text{ (modulo 11)}$$
$$10 \times c_1 + 1 \times c_2 + 9 \times c_3 + ... + 1 \times c_{20} = 0 \text{ (modulo 11)}$$

In other words, a string $\mathbf{c}=(c_1,c_2,....,c_{20})$ with elements from $\mathbb{Z}_{11}$ is a codeword if, and only if, it has inner product zero (modulo 11) with the rows of the following matrix **H3**.

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}	n_{13}	n_{14}	n_{15}	n_{16}	n_{17}	n_{18}	n_{19}	n_{20}
	1	0	1	0	1	0	1	0	1	0	1	2	3	4	5	7	8	1	0	0
H3	0	1	0	1	0	1	0	1	0	1	0	1	2	3	4	6	7	0	1	0
	10	1	9	2	8	3	7	4	6	5	4	5	7	10	3	2	8	0	0	1

Error detection simply takes place by checking if the received word $\mathbf{r}=(r_1,r_2,...,r_{20})$ satisfies the three parity check equations.

Encoding can for example be done as follows: choose c_1, c_2,..., c_{17} in any way. If we define

$$c_{18} = -(1 \times c_1 + 0 \times c_2 + 1 \times c_3 + ... + 8 \times c_{17}) \text{ modulo 11}$$
$$c_{19} = -(0 \times c_1 + 1 \times c_2 + 0 \times c_3 + ... + 7 \times c_{17}) \text{ modulo 11}$$
$$c_{20} = -(10 \times c_1 + 1 \times c_2 + 9 \times c_3 + ... + 8 \times c_{17}) \text{ modulo 11}$$

then (c_1, c_2,..., c_{20}) is a codeword. We can view c_{18}, c_{19} and c_{20} as parity check digits.

NOTE 1 We may restrict c_1, c_2,..., c_{17} to be any of the numbers 0, 1, 2..., 9. Any of the three parity check digits can be '10'. This '10' can be represented by an alphanumerical character different from 0, 1,..., 9, for example X or Z.

Decoding is done by

$$s_{18} = (1 \times c_1 + 0 \times c_2 + 1 \times c_3 + ... + 1 \times c_{18}) \text{ modulo 11}$$
$$s_{19} = (0 \times c_1 + 1 \times c_2 + 0 \times c_3 + ... + 1 \times c_{19}) \text{ modulo 11}$$
$$s_{20} = (10 \times c_1 + 1 \times c_2 + 9 \times c_3 + ... + 1 \times c_{20}) \text{ modulo 11}$$

In short, the code defined with **H3** detects all errors of any of the following types.

- Single and double substitution errors.
- Single and double transposition errors.

- Any combination of a single substitution error and a single transposition error.
- All three consecutive substitution errors.

where a transposition is ab => ba and a substitution is a => b.

EXAMPLE The following example illustrates the use of the algorithm on valid UDN as input number.

Position (n)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Input number	**8**	**5**	**6**	**2**	**8**	**7**	**0**	**1**	**2**	**1**	**5**	**3**	**2**	**9**	**5**	**6**	**7**				Choose a digit (0..9)

Matrix **H3**																				
	1	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**2**	**3**	**4**	**5**	**7**	**8**	**1**	**0**	**0**	Row for c_{18} & s_{18}
	0	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**2**	**3**	**4**	**6**	**7**	**0**	**1**	**0**	Row for c_{19} & s_{19}
	10	**1**	**9**	**2**	**8**	**3**	**7**	**4**	**6**	**5**	**4**	**5**	**7**	**10**	**3**	**2**	**8**	**0**	**0**	**1**	Row for c_{20} & s_{20}

Coding checkdigit = -sum($n_1..n_{17}$) mod 11

c_{18}	8	0	6	0	8	0	0	0	2	0	5	6	6	36	25	42	56				***9***
c_{19}	0	5	0	2	0	7	0	1	0	1	0	3	4	27	20	36	49				***10***
c_{20}	80	5	54	4	64	21	0	4	12	5	20	15	14	90	15	12	56				***2***
Codeword	**8**	**5**	**6**	**2**	**8**	**7**	**0**	**1**	**2**	**1**	**5**	**3**	**2**	**9**	**5**	**6**	**7**	**9**	**10**	**2**	

Decoding checkdigit = +sum($n_1..n_{17}$ and n_{18} or n_{19} or n_{20}) mod 11

s_{18}	8	0	6	0	8	0	0	0	2	0	5	6	6	36	25	42	56	9	0	0	***0***
s_{19}	0	5	0	2	0	7	0	1	0	1	0	3	4	27	20	36	49	0	10	0	***0***
s_{20}	80	5	54	4	64	21	0	4	12	5	20	15	14	90	15	12	56	0	0	2	***0***

NOTE 2 The value '10' of checksum digit c_{19} can be represented by an alphanumerical character different from {0, 1,..., 9}, for example X or Z.

A.9.3 ARC checksum – Definition

The checksum on the ARC is calculated by F-ARC.

Take n = 12, r = 2 and p = 11. We consider the code defined by the r = 2 following check equations.

$$8 \times c_1 + 8 \times c_2 + 6 \times c_3 + ... + 1 \times c_{11} = 0 \text{ (modulo 11)}$$
$$3 \times c_1 + 6 \times c_2 + 4 \times c_3 + ... + 1 \times c_{12} = 0 \text{ (modulo 11)}$$

In other words, a string (c_1, c_2,..., c_{12}) with elements from $\mathbb{Z}_{11}$ is a codeword if and only if it has inner product zero (modulo 11) with both rows of the following matrix **H1**:

H1	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}
	8	8	6	5	10	5	6	4	1	4	1	0
	3	6	4	2	6	8	2	1	2	4	0	1

Error detection simply takes place by checking if the received word $\mathbf{r} = (r_1, r_2, ..., r_{12})$ satisfies the two parity check equations.

Encoding can for example be done as follows: choose c_1, c_2,..., c_{10} in any way. If we define

$$c_{11} = -(8 \times c_1 + 8 \times c_2 + 6 \times c_3 + ... + 4 \times c_{10}) \text{ modulo 11}$$
$$c_{12} = -(3 \times c_1 + 6 \times c_2 + 4 \times c_3 + ... + 4 \times c_{10}) \text{ modulo 11}$$

then (c_1, c_2,..., c_{12}) is a codeword. We can view c_{11} and c_{12} as parity check digits.

NOTE We may restrict c_1, c_2,..., c_{10} to be any of the numbers 0, 1, 2..., 9. Any of the two parity check digits can be '10'. This '10' can be represented by an alphanumerical character different from 0, 1,..., 9, for example X or Z.

Decoding is done by:

$$c_{11} = (8 \times c_1 + 8 \times c_2 + 6 \times c_3 + ... + 1 \times c_{11)} \text{ modulo } 11$$
$$c_{12} = (3 \times c_1 + 6 \times c_2 + 4 \times c_3 + ... + 1 \times c_{12}) \text{ modulo } 11$$

From this table, we draw the following conclusions.

- All single and double substitution errors are detected.
- All single and double transposition errors are detected.
- Any combination of a substitution error in position 12, and a transposition error in positions not involving position 12 is detected.
- A substitution error not in position 12 "matches" exactly one transposition error. About 1 % not detected.

where a transposition is ab => ba and a substitution is a => b.

EXAMPLE The following example illustrates the use of the algorithm on valid ARC as input number.

Position (n)	1	2	3	4	5	6	7	8	9	10	11	12	
Input number	1	6	6	0	8	7	3	1	0	1			Choose a digit (0..9)
Matrix **H1**	8	8	6	5	10	5	6	4	1	4	1	0	Row for c_{11} & s_{11}
	3	6	4	2	6	8	2	1	2	4	0	1	Row for c_{12} & s_{12}

Coding checkdigit = -sum($n_1..n_{10}$) mod 11

c_{11}	8	48	36	0	80	35	18	4	0	4		***9***
c_{12}	3	36	24	0	48	56	6	1	0	4		***9***
Codeword	1	6	6	0	8	7	3	1	0	1	9	9

Decoding checkdigit = +sum($n_1..n_{10}$ and n_{11} or n_{12}) mod 11

s_{11}	8	48	36	0	80	35	18	4	0	4	9	0	***0***
s_{12}	3	36	24	0	48	56	6	1	0	4	0	9	***0***

A.10 Use of PKCS#1

A.10.1 RSA signatures under PKCS#1

RSA signatures shall be made as specified by the implementation guidelines of IETF RFC 3447.

The scheme is RSA + SHA1. There are two choices specified in IETF RFC 3447 as they are RSASSA-PSS and RSASSA-PKCS1-V1_5.

Since OMA DRM 2.0 is used for interactive mode of operation and uses RSASSA-PSS, RSASSA-PSS shall be used to sign the binary messages for broadcast mode of operation.

A.10.2 RSA encryption/decryption under PKCS#1

RSA encryption/decryption shall be done as specified by the implementation guidelines of IETF RFC 3447.

The recommended cipher in IETF RFC 3447, RSAES-OAEP, shall be used for RSA encryption and decryption.

A.11 Tag length format for keyset_block

A.11.1 TLF syntax definition

A tag length format (TLF) is defined to identify the keyset_items in the keyset_block. A keyset_item is identified by following syntax:

<tag> [optional <clarifier>] [optional <length>] <keyset_item>

The following values are defined and shall be used.

A.11.2 Tag

This is a 4-bit field (bslbf) indicating the tag that uniquely identifies the keyset item, as specified in Table A.8.

Table A.8 – Defined tag values

Keyset_item	Tag (b)	Clarifier present	Remark
UGK	0000	NO	
SGK	0001	YES	
UDK	0010	NO	
UDF	0011	NO	
BDK	0100	NO	
SBDF	0101	NO	shortform_domain_id
LBDF	0110	YES	
RIAK	0111	NO	
TDK	1000	NO	
Reserved for future use.	1001-1101	YES	Not used in this standard
TAA	1110	YES	For its specification, see A.11.7 If present in a keyset block, the TAA shall be the first keyset item.
CGF	1111	YES	See 7.2.2

- The keyset items shall be included in the order of the table above, except for TAA, which shall be the first keyset item if it is present.
- The keyset shall include at most one instance of each the following keys: UGK, UDK, UDF, RIAK and TDK.
- If included the SGKs (8 or 9) shall follow in fashion SGK1..n.
- The keyset may include zero or more domain sets (BDK, SBDF, LBDF). If included, the SBDF shall follow the BDK it belongs to, followed by the optional LBDF that belongs to the afore-mentioned SBDF.

A.11.3 Clarifier (optional)

This 10-bit field (bslbf) can be used to indicate the following possible values. See also Table A.8 for the presence and absence of the clarifier field.

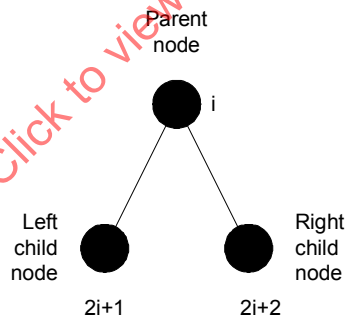
- In case the preceding <tag> value indicates a SGK, this field shall be present. It shall represent the position of a SGK in the fiat naor tree.

- In case the preceding <tag> value indicates a LBDF, this field shall be present. It shall represent the length of the LBDF in bytes and be of type uimsbf.
- In case the preceding <tag> value indicates a TAA, this field shall be present. It shall represent the length of the next keyset_item field in bytes and be of type uimsbf.
- In case the preceding <tag> value indicates a CGF, this field shall be present. It shall represent the length of the CGF in bytes and be of type uimsbf.
- Future extensions to this standard shall always use tag + clarifier + length + keyset_item. The clarifier field shall indicate the length of the keyset_item field in bytes and be of type uimsbf.
- In all other cases, the clarifier field shall not be present.

A.11.4 Specifying the use of the clarifier field for position of SGK

A.11.4.1 General

If keyset_item == 0001 (i.e. SGK) then the field "clarifier" shall indicate the position of the SGK as a node in the fiat naor tree (see Fiat Naor:1998). When m = group size, then n = 2 log (m), where n is number of SGKs in tree. Possible positions for the SGKs in the tree are $2^{(n+1)} - 1$. Therefore, the parameter "position" is expressed with 10 bits to express 1 023 nodes in a tree. The first most significant bit left will be used as binary indicator to indicate if the SGK position is a node (0, zero) or a leaf (1, one). Bit positions 2..10 (from left to right lsb) are used in binary format as an indication of the node and leaf position. Nodes shall be numbered according to Figure A.7. The nodes at the bottom level of the tree (for example, D0 in Figure A.5) are also called leaves. Leaves represent devices. A leaf is a node (for example, D0 == NK7 in Figure A.5), but a node does not need to be a leaf (for example, NK5 in in Figure A.5). Leaves are numbered from left to right, starting at 0. See in Figure A.5 for examples.



Key

The root key R is numbered zero. Node keys NK are sequentially numbered per "level" in a breadth-first manner from left to right, starting from the root node with number 0.

Figure A.7 – Node numbering

A.11.4.2 Specifying the use of the clarifier for length of LBDF

If LBDF is included, the field "clarifier" specifies the variable length of the LBDF in bytes, as specified in A.11.6.

A.11.4.3 Specifying the use of the clarifier for length of TAA

If the TAA is included, the field "clarifier" specifies the variable length of the TAA in bytes.

A.11.4.4 Specifying the use of the clarifier for length of CGF

If CGF is included, the field "clarifier" specifies the variable length of the CGF in bytes, as specified in 7.2.2.4 and 9.3.2.7.2.

A.11.4.5 Specifying the use of the length field

This is a 3-bit field (bslbf) indicating the length of a keyset item, as specified in Table A.9 and Table A.10. This field shall be present for all keyset items, except for the LBDF keyset item.

Table A.9 – Defined length values

(Key) length prescriber	Length (b)	Remark
128 bit AES	000	
192 bit AES	001	
256 bit AES	010	
5 byte Eurocrypt	011	
6 byte	100	SBDF
Reserved for future use	101-110	Not used in this standard
nil field	111	
NOTE The nil field may be used in future extensions to ensure that future codes can be parsed by legacy devices.		

Table A.10 – Correct usage of length values

Keyset_item	Allowed (key) length prescriber (binary)
UGK	000, 001 or 010
SGK	000, 001 or 010
UDK	000, 001 or 010
UDF	011
BDK	000, 001 or 010
SBDF	100
LBDF	Shall use clarifier field for length
RIAK	000, 001 or 010
TDK	000, 001 or 010
TAA	nil (111)
CGF	Shall use clarifier field for length

A.11.5 TLF examples

EXAMPLE 1 A 5-byte Eurocrypt address implementing the UDF will be coded as

```
<0011> <011> <UDF>
```

EXAMPLE 2 A 48 bits SBDF address will be coded as

```
<0101> <100> <SBDF>
```

EXAMPLE.3 A LBDF address of 105 bytes will be coded as

```
<0110> <0001101001> <LBDF>
```

EXAMPLE 4 A 128 but AES key implementing the UGK will be coded as

```
<0000> <000> <UGK>
```

EXAMPLE 5 A 128-bit AES key implementing the SGK on node position NK5 in Figure A.5 will be coded as

```
<0001> <0000000101> <000> <SGK>
```

EXAMPLE 6 A 128-bit AES key implementing the SGK on node position NK7 (i.e. D0) in Figure A.5 will be coded as

```
<0001> <1000000000> <000> <SGK>
```

EXAMPLE 7 A CGF of 10 bytes will be coded as

```
<1001> <0000001010> <CGF>
```

A.11.6 LBDF syntax

In OMA DRM 2.0, the domain ID can be 1 to 17 characters (any) followed by 3 digit characters.

The string that forms the identifier is encoded normally in ROAP messages using UTF-8. The UTF-8 character encoding for ASCII characters is 'efficient' with 1 byte per character. On the other hand, there are characters that are encoded using 6 bytes (Asian languages).

The 17 XML UTF-8 characters shall be translated into bytes as follows.

The longest OMA DRM 2.0 domain identifier encoded as bytes is $6 \times 17 + 3$ bytes = 105 bytes.

The shortest domain identifier is 4 bytes.

A.11.7 TAA descriptor syntax

The syntax of the TA Algorithm TLF fields is specified in Table A.11.

Table A.11 – TAA descriptor syntax

Field	Length	Value	Type
TAA_descriptor() {			
tag	4	1110_b	bslbf
clarifier	10		uimsbf
length	3	111_b	bslbf
TAA_TA_ID	10		uimsbf
TAA_report_value	56		uimsbf
TAA_algorithm	8		uimsbf
TAA_parameter	8*clarifier – 74 – n		bslbf
TAA_padding	n (n>0)	0_b (n times)	bslbf
}			

tag – This is the tag for TAA as defined in Table A.8.

clarifier – This is the clarifier for TAA as defined in A.11.1.

TAA_TA_ID – This field contains the Identifier of the trust authority that defined the algorithms for extra encryption of the keyset block in which this descriptor is located; see 9.3.2.7.2 and 9.3.3.

TAA_report_value – This field contains the value to be used in the determination of the TAA report to the RI of the reception of the TAA in the registration message; see 9.3.2.3.8.

TAA_algorithm – This field contains which algorithm is used for extra encryption of the keyset block; see 9.3.2.7.2 and 9.3.3. See Table A.12 for the values of this field and their meaning.

Table A.12 – TAA algorithm values

Field value	Meaning	Remark
0x00-0xff	Indicates the extra encryption algorithm of the TA indicated by the field TAA_TA_ID	The algorithms are defined by the TA

TAA_parameter – This field contains the parameter(s) for the algorithm indicated by the TAA_algorithm field. The parameter(s) are defined by the trust authority indicated in the TAA_TA_ID field.

TAA_padding – This field is an n-bit zero-valued padding field. The total length of the TAA tag field, clarifier field, length field and keyset_item field shall be 1 bit plus a multiple (m) of 64 bits. There shall be at least 1 padding bit, i.e. 17 + clarifier × 8 = m × 64 + 1. There shall be at least one padding bit, because the last bit of the TAA_parameter field is encrypted with the TAA encryption; see 9.3.2.7.2 and 9.3.3.

A.12 Message_tag overview

The messages that are defined in this standard shall use message_tag values as specified in Table A.13.

Table A.13 – Message_tag overview

Message name	Message_tag	Specified in
BCRO()	0x20	8.4.2
device_registration_response()	0x01	9.3.2.7.2
re_register_msg()	0x11	9.3.2.7.3
update_ri_certificate_msg()	0x12	9.3.2.7.4
update_drmtime_msg()	0x13	9.3.2.7.5
update_contact_number_msg()	0x14	9.3.2.7.6
domain_registration_response()	0x02	9.3.3.5.1
domain_update_response()	0x03	9.3.3.5.2
join_domain_msg()	0x15	9.3.3.5.3
leave_domain_msg()	0x17	9.3.3.5.4
present_TAA_report()	0x18	9.3.2.7.7
token_delivery_response()	0x30	9.3.4.4

A.13 Table ID overview

Table A.14 lists all table IDs that are used in the MPEG SI tables used in this standard.

Table A.14 – Table ID overview

Table ID	Value	Stream	Remark
KSM_TABLE80_ID	0x080	KSM	To indicate change in data, for example, odd/ even content key
KSM_TABLE81_ID	0x081	KSM	To indicate change in data, for example, odd/ even content key
BCRO_TABLE_ID	BaseTableID+0x02	RI service	
DEVICE_REGISTRATION_TABLE_ID	BaseTableID+0x03	RI service	
RE_REGISTRATION_TABLE_ID	BaseTableID+0x04	RI service	
UPDATE_RI_CERTIFICATE_TABLE_ID	BaseTableID+0x05	RI service	
UPDATE_DRMTIME_TABLE_ID	BaseTableID+0x06	RI service	
UPDATE_CONTACT_NUMBER_TABLE_ID	BaseTableID+0x07	RI service	
DOMAIN_REGISTRATION_TABLE_ID	BaseTableID+0x08	RI service	
DOMAIN_UPDATE_TABLE_ID	BaseTableID+0x09	RI service	
JOIN_DOMAIN_TABLE_ID	BaseTableID+0x0A	RI service	
LEAVE_DOMAIN_TABLE_ID	BaseTableID+0x0B	RI service	
TOKEN_DELIVERY_TABLE_ID	BaseTableID+0x0C	RI service	
PURCHASE_CHANNEL_TABLE_ID	BaseTableID+0x0D	Stream referenced to by purchase_info_descript or or CA_PID in CA_descriptor_loop	
SERVICE_BUNDLE_TABLE_ID	BaseTableID+0x0E	Stream referenced to by purchase_info_descript or or CA_PID in CA_descriptor_loop	
PURCHASE_ITEM_TABLE_ID	BaseTableID+0x0F	Stream referenced to by purchase_info_descript or or CA_PID in CA_descriptor_loop	

Table A.14 lists the table IDs used for the SI-tables defined in this standard. The BaseTableID is 0xB0. All table IDs are in the user-defined range. Without any knowledge of how the table is signalled or on which special stream the table is transmitted, a terminal will not be able to uniquely identify a table as other specifications might define tables with the same IDs. It is therefore important that PIDs carrying tables defined in this standard shall not carry other tables with the same table IDs.

A.14 Multilingual text structure

The multilingual text structure is used to convey text in different languages. Wherever multilingual_text() is used, this is replaced with the structure given by Table A.15.

Table A.15 – Multilingual text structure

Multilingual text structure	Length	Type
multilingual_text() {		
ISO_639_language_code	24	bslbf
text_length	8	uimsbf
international_text	8*text_length	bslbf
}		

ISO_639_language_code – This 24-bit field identifies the language of the text in the 'international_text' field. The ISO_639_language_code contains a 3-character code as specified by ISO 639-2:1998. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is coded into 8 bits according to ISO/IEC 8859-1:1998 and inserted in order into the 24-bit field.

EXAMPLE: English has the 3-character code 'eng', which is coded as: "0110 0101 0110 1110 0110 0111".

text_length – This field specifies the length of the following text in bytes.

international_text – Variable length field with length text_length*8 bits. This field contains the text information. The encoding of the text and the character set used is specified in ETSI EN 300 468, Annex A.

A.15 Authentication of the tokens_consumed field in the token consumption data

Devices shall authenticate the tokens_consumed field and the device_nonce of the token consumption report; see 9.3.2.3.5 in the way specified in this clause. The hash function used here is one of the four secure hash functions from Schneier:1996, page 449 (the upper left one in Figure 18.9).

The maximum amount of tokens that can be reported as consumed is 9999. The amount of tokens, with the above-mentioned restriction, is represented with a 14-bit uimsbf number and called tokens_consumed. The value of device_nonce can be in value between 0 and 9 and is represented as a 4-bit uimsbf number. The 14-bit number tokens_consumed is right concatenated with the 4-bit number device_nonce. The resulting 18-bit number is right padded with 0x1 and right padded again with 109 binary zeroes (so 2^{109}). The resulting 128-bit number is used as the input for a single AES block. The report authentication key, as obtained with the token delivery response message (see 9.3.4.4) is used as the key input for the AES block. The 128-bit output of the AES block is EXOR-ed with the 128-bit input of the AES block. The left-most 43 bits of the result of this EXOR operation are taken as the report_authentication_code. The 13-digit decimal representation of these 43 bits, including

any leading zeroes is used as the report_authentication_code in the token consumption report. See also Figure A.8.

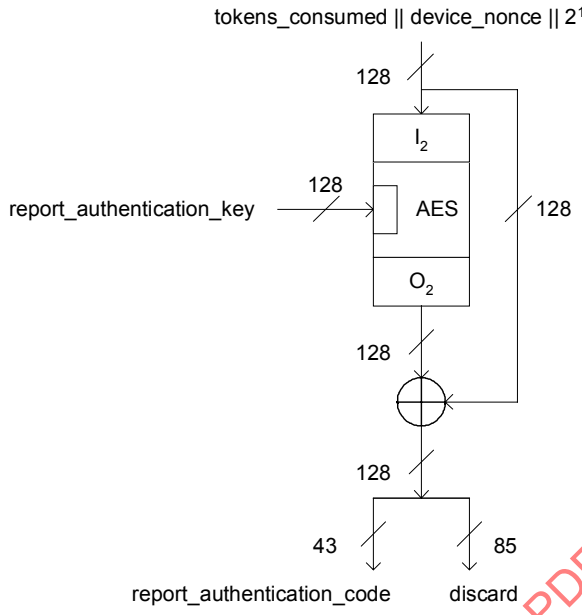


Figure A.8 – Computation of the report_authentication_code

A.16 Management of tokens by RIs and devices

A.16.1 Token management by RIs

A.16.1.1 General

There are two business models for the use of tokens.

The first business model is that all tokens ordered by the user are paid for by the user. These tokens are pre-paid tokens. The second business model is that a user orders tokens, but only wants to pay for the ones he actually consumes. These tokens are post-paid tokens.

The tools to support these two business models are the token delivery response message (see 9.3.4.4) and the token reporting protocol (see 9.3.4.3). The next subclauses specify how these tools may be used by an RI to support the above two business models and how one can switch from one business model to the other.

A.16.1.2 Pre-paid token business model

Setting the token_reporting_flag in the token delivery response message to 0x0 will signal to the device that it does not now nor in the future have to report anymore on the consumption of any of the tokens received so far from this RI.

Therefore, in the pre-paid token business model, where the user has agreed to be billed for the delivery of the tokens and their consumption need not be reported, the RI will set the token_reporting_flag to 0x0.

Tokens that are delivered from an RI to a device with a token delivery response message which token_reporting_flag has been set to 0x0 can be called pre-paid tokens.

A.16.1.3 Post-paid token business model

Setting the token_reporting_flag in the token delivery response message to 0x1 will signal to the device that it shall report on the consumption of these tokens.

NOTE 1 Although a broadcast device can only display the token consumption message to the user and relies on the user to report this message to the RI, the wording in this subclause is as if the device does the reporting.

Therefore, in the post-paid token business model, where the user has to be billed for the actual consumption of the tokens and their actual consumption shall be reported by the device, the RI will set the token_reporting_flag to 0x1.

Tokens that are delivered from an RI to a device with a token delivery response message which token_reporting_flag has been set to 0x1 can be called post-paid tokens.

In the post-paid token business model, the RI can limit its risk, by making sure that a device at all times only contains post-paid tokens up to a certain maximum, the so-called credit-limit. Furthermore, the RI can set a date/time limit in the device after which the device is not allowed to consume post-paid tokens any more. This can be done as follows.

a) The RI sends in the first token delivery response message a number of tokens equal to the credit limit. Furthermore, the RI sets the token_reporting_flag in the token delivery response message to 0x1 and sets the latest_consumption_time to a suitable date/time.

b) The RI waits for the reception of a token consumption message.

c) If the RI receives a token consumption message, it shall check the authenticity of the tokens_consumed field. If the authentication fails, go to step b); otherwise, continue with step d).

d) For reasons explained in A.16.1.4, if the reported number of consumed tokens is higher than the credit limit, the RI shall assume that only a number of post-paid tokens equal to the credit-limit have been consumed by the device.

e) The RI bills the user for the amount of post-paid tokens consumed with a maximum equal to the credit-limit.

f) The RI sends a token delivery response message a number of tokens equal to the amount of post-paid tokens consumed with a maximum equal to the credit limit. Furthermore, the RI sets the token_reporting_flag in the token delivery response message to 0x1 and sets the latest_consumption_time to a suitable date/time.

g) Go to step b).

NOTE 2 In the above, the use of the response_flag, device_nonce, earliest_reporting_time, latest_reporting_time and other fields has not been included.

NOTE 3 The RI can force the creation of a token consumption message by sending a token delivery response message with its status field set to "TokenConsumptionMessageError" or to "NoTokenConsumptionMessage".

A.16.1.4 Switching from the pre-paid token business model to the post-paid token business model

When at a certain point of time, the user asks the RI to switch from the use of pre-paid tokens to the use of post-paid tokens and the RI agrees, the RI starts at step a) in A.16.1.3. The device will report the actual consumption of tokens that have been delivered to it in step a) and in all steps f). However, at the time of executing step a), the device may still have some pre-paid tokens. Based on the implementation of the device, these pre-paid tokens may also be reported as consumed by the device; see A.16.2. Because the RI knows that a device never holds more post-paid tokens than the credit limit, the RI shall assume that at most an amount of tokens equal to the credit limit have been consumed by the device. Hence step d) in A.16.2.

A.16.1.5 Switching from the post-paid token business model to the pre-paid token business model

When at a certain point of time, the user asks the RI to switch from the use of post-paid tokens to the use of pre-paid tokens, and the RI agrees, the RI has a few options.

One option is that the RI bills the user for the amount of post-paid tokens that were left in the device at the time of the last token consumption message. These tokens have in effect then become pre-paid tokens. The RI will set the token_reporting_flag in the next token delivery response message to 0x0 and set the value of token_quantity to zero or to the amount of tokens that the user wished to purchase in addition to the amount of post-paid tokens left in the device (encrypting token_quantity yields the encrypted_token_quantity field).

Another option, useful, for example, when the previous option turns out to be expensive, is that the RI performs the actions specified in A.16.1.6 for clearing the post-paid tokens left in the device. After clearing the post-paid tokens, the RI can start sending pre-paid tokens to the device if the user wishes to purchase these.

A.16.1.6 Stopping the post-paid token business model

When at a certain point of time, the user informs the RI that he no longer wishes to use post-paid tokens, not even the ones that are still in his device, the RI can do the following. The RI sends the device a token delivery response message with

- the value of token_quantity set to zero (encrypting token_quantity yields the encrypted_token_quantity field) or set the token_quantity_flag to 0x0;
- the token_reporting_flag field set to 0x1;
- the latest_token_consumption_time set to a date/time in the past;
- the status field to "NoTokenConsumptionMessage".

This forces the device to generate a token consumption report. Using this, the RI determines how many post-paid tokens are still in the device. The RI sends the device a token delivery response message with

- the value of token_quantity set to minus the amount of post-paid tokens left in the device (encrypting token_quantity yields the encrypted_token_quantity field);
- the token_reporting_flag field set to 0x1;
- the latest_token_consumption_time set to a date/time in the past;
- the status field to "Success".

The above message may be repeated several times. After reception of this token delivery message, the device will have no post-paid tokens left. Any remaining pre-paid tokens can still be consumed by the device.

A.16.2 Token management by devices

Each RI context in a device shall at a minimum contain

- a token purse, which is incremented with the tokens received from the RI and which is decremented with the amount of tokens required for each requested consumption of metered protected content from the corresponding RI. If a decrement would yield an accumulator value of less than zero, the device shall deny the requested consumption of metered protected content from the corresponding RI;
- a token consumption accumulator, which initially starts at zero;
- the token purse initially starts at zero.

Whenever a device receives from an RI a token delivery response message that has its token_reporting_flag set to 0x0, the device sets the token consumption accumulator associated with the RI to zero and shall consider all tokens in the token purse associated with the RI as pre-paid tokens.

In case of the reception of a (series of) token delivery response message with the token_reporting_flag set to 0x1, there are two possible implementations of token consumption registration.

A device with a simple implementation of token consumption registration would do the following.

a) Whenever tokens are required and available for consumption, then, in addition to decrementing the token purse associated with the RI, the device also increments the token consumption accumulator associated with the RI with the same amount.

b) When a device receives a token delivery response message with status "Success" and a token_reporting_flag set to 0x1, the device shall schedule the creation of a token consumption report at a suitable time, keeping in mind the value in the field latest_consumption_time and possibly the values in the fields earliest_reporting_time and latest_reporting_time.

   If the response_flag was set to 0x1, the device shall decrement the token consumption accumulator associated with the RI with the amount of tokens reported in the token consumption report that this token delivery response message was a response to. The device may delete that token consumption report and all others sent before that token consumption report was sent.

   The device shall increment the token purse associated with the RI with the number of tokens indicated in the encrypted_token_quantity field of this token delivery response message.

c) When a device receives a token delivery response message with status "TokenConsumptionMessageError" or with status "NoTokenConsumptionMessage", the device shall immediately create a token consumption report.

d) When a device creates a token consumption report, it uses a device_nonce that is one higher than the previously used device_nonce. If the previously used device_nonce was 9, the device uses 0 as the next device_nonce.

e) When a device creates a token consumption report, it uses the value of the token consumption accumulator associated with the RI as the amount of tokens to report as consumed.

f) Token consumption reports shall be stored by the device, at least until the device receives a token delivery response message indicating that they have been successfully been processed by the RI or indicating that later created token delivery messages have been successfully been processed by the RI.

g) The device shall stop with the execution of the above actions when it receives a token delivery response message with the token_reporting_flag set to 0x0. The device shall then set the token consumption accumulator associated with the RI to zero and may delete all created token consumption reports.

The device actions above imply that the RI will consider the first credit_limit tokens reported as consumed to be post-paid tokens, even though the device might still have had pre-paid tokens. Furthermore, if the current date/time is past the date/time set in the latest_token_consumption_time field, a device is not allowed to use tokens any more. Since a device according to the above rules does not keep track separately of the pre-paid tokens, it cannot use tokens anymore even though the device might still have had pre-paid tokens.

With a slightly more complex implementation, the above two disadvantages can be solved. The device can then first consume all pre-paid tokens before consuming any post-paid tokens. To this end, the device would implement two token purses per RI, a pre-paid token purse and a post-paid token purse for tokens that have been delivered to the device with token delivery

response messages with the token_reporting_flag set to 0x1. The device can first consume all tokens in the pre-paid token purse. Consumption of tokens from the pre-paid token purse will not influence the value of the token consumption accumulator and this consumption will therefore not be reported by the device. Whenever the device consumes tokens from the other token purse, the token consumption accumulator shall be incremented with the same amount. A device according to this implementation, upon the reception of a token delivery response message with the token_reporting_flag set to 0x0, shall increment the pre-paid token purse with the tokens in the post-paid token purse, shall set the post-paid token purse as well as the token consumption accumulator to zero.

A.17 Service guide and CDP signalling for IPDC over DVB-H systems

A.17.1 General

This standard requires certain information to be signalled or carried in the service guide, alongside protected services. This clause specifies the formatting of this data in the ESG and CDP for IPDC over DVB-H systems.

The ESG is fully specified by

- the general structure as in ETSI TS 102 471;
- the data formats relating to this standard as in this clause;
- the data field definitions as in 12.5.

The CDP is fully specified in ETSI TS 102 472, with the addition of the extra parameter in the SDP as specified in A.17.5.

A.17.2 Data mapping

A.17.2.1 Signalling of service operation centre

All service information centre data is included in the PurchaseChannel fragment and in the private data element, using ServiceOperationCentreType.

```
<complexType name="ServiceOperationCentreType">
    <complexContent>
        <extension base="esg:PrivateDataType">
            <sequence>
                <element name="socID" type="anyURI"/>
                <element name="socName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
                <element name="socKeyURL" type="anyURI"/>
                <element name="socInfoURL" type="anyURI"/>
                <element name="socRiProxyURL" type="anyURI" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

In order to signal the socID using a XML type of anyURI, the actual socID is concatenated to the following URI:

"http://www.18crypt.com/socID:".

Given a socID of, for example, 1234 the socID SHALL be signalled as follows:

<c18:socID>http://www.18crypt.com/socID:1234</c18:socID>

A.17.2.2 Signalling of customer operation centre

All customer operation centre data is included in the PurchaseChannel fragment and in the private data element, using CustomerOperationCentreType.

```
<complexType name="CustomerOperationCentreType">
    <complexContent>
        <extension base="esg:PrivateDataType">
            <sequence>
                <element name="cocID" type="anyURI"/>
                <element name="cocName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
                <element name="cocLocalFlag" type="boolean"/>
                <element name="cocRekeyingSafetyWindow" type="integer"/>
                <element name="cocPurchaseURL" type="anyURI" minOccurs="0"/>
                <element name="cocPortalURL" type="anyURI" minOccurs="0"/>
                <element name="cocContactInfo" type="mpeg7:TextualType"/>
                <element name="cocRiID" type="roap:Identifier"/>
                <element name="cocRiURL" type="anyURI"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

A.17.2.3 Signalling of service

Service information is given in the Service fragment. Service id, name and description are signalled with the respective fields in the service fragment. ServiceTypeEnum is signalled with the ServiceType element with ServiceTypeCS.

The serviceBaseCID is signalled in the private data element, using HorizontalServiceInformationType.

```
<complexType name="HorizontalServiceInformationType">
    <complexContent>
        <extension base="esg:PrivateDataType">
            <sequence>
                <element name="serviceBaseCID" type="string"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

When serviceBaseCID is used to construct a type that is a CID, e.g. service_CID or programme_CID in 7.2.4, and if serviceBaseCID starts with a prefix of the form “scheme:”, with ‘scheme’ as defined in IETF RFC 1738, e.g. cid:, the prefix shall be dropped when constructing the CID, since CIDs cannot contain more than one prefix.

A.17.2.4 Signalling of ScheduleItem

ScheduleItem information is given in the ScheduleEvent fragment, except for the name and description that are given in the associated Content fragment. The schedule item id shall be instantiated.

A.17.2.5 Signalling of PurchaseItem and PurchaseData

PurchaseItem information is given in the Purchase fragment, using HorizontalPurchaseItemType. The purchase item name, version and id are given in pItemName, pItemVersion and pItemID fields, respectively. The purchase fragment has a reference to a ServiceBundle fragment, which is the sellable item.

pItemServiceId, pItemScheduleId and pItemContentId are not part of this fragment. Instead, service can be sold by instantiating a ServiceBundle fragment that refers just to that service. A scheduleItem can be sold by instantiating a service that contains just one scheduleItem, and selling that service.

SubscriptionTypeEnum values are: 1 for continuous subscription, 2 for one-time subscription, and 3 for preview.

```
    <simpleType name="subscriptionTypeEnumType">
        <restriction base="integer">
            <enumeration value="1"/>
            <enumeration value="2"/>
            <enumeration value="3"/>
        </restriction>
    </simpleType>
    <complexType name="HorizontalPurchaseItemType">
        <complexContent>
            <extension base="esg:PurchaseDataBaseType">
                <sequence>
                    <element name="pItemName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
                    <element name="pItemVersion" type="integer"/>
                    <element name="purchaseOption">
                        <complexType>
                            <sequence>
                                <element name="SubscriptionUnit" nillable="true"
minOccurs="0">
                                    <complexType>
                                        <attribute name="subscriptionTypeEnum"
type="c18:subscriptionTypeEnum" use="required"/>
                                        <attribute name="subscriptionDuration" type="duration"
use="required"/>
                                    </complexType>
                                </element>
                                <element name="UnitText" type="mpeg7:TextualType"
minOccurs="0" maxOccurs="unbounded"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
                <attribute name="pItemId" type="unsignedInt" use="required"/>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name="subscriptionTypeEnum">
        <union memberTypes="c18:subscriptionTypeEnumType integer"/>
    </simpleType>
```

A.17.3 Rights issuer services

A.17.3.1 Signalling of rights issuer services

Rights issuer services are signalled using the ServiceType scheme. The ServiceType element is used to identify the service as an RI service by referencing 'RI service' in the ClassificationScheme urn:dvb:ipdc:esg:cs:ServiceTypeCS. Additional data for the RI service is signalled using the RIServiceDataType that is used as an instance of the PrivateData element in ServiceType.

```
<complexType name="RIServiceDataType">
<complexContent>
        <extension base="esg:PrivateDataType">
            <sequence>
                <element name="riID" type="roap:Identifier"/>
                <element name="scheduled" type="boolean"/>
                <element name="drmID" type="anyURI" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

A.17.3.2 Signalling of scheduled rights issuer service schedule

Rights issuer services can be addressed to certain groups or devices. In order to allow this the PrivateData element of the ContentType is used to add group addressing. Scheduling of these ContentTypes is done by using the ScheduleEventType in the usual way.

```
<complexType name="RIAddressingType">
    <complexContent>
        <extension base="esg:PrivateDataType">
            <sequence>
                <element name="EuroCryptGroup" type="unsignedLong"/>
                <element name="EuroCryptGroupMask" type="unsignedLong"/>
                <element name="DeviceAddress" type="unsignedByte" minOccurs="0"/>
                <element name="DeviceAddressMask" type="unsignedByte" minOccurs="0"/>
                <element name="CertificateChainUpdate" type="boolean"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

The fields in the RIAddressingType define the range of devices to be addressed in this timeslot. A device should evaluate the EuroCryptGroup and EuroCryptGroupMask as described below to determine whether its EuroCrypt Group will be addressed in this timeslot. If its group will be addressed, the device should then evaluate the DeviceAddress and DeviceAddressMask fields to determine whether it will be addressed in this timeslot. If it will, the device should listen to the Rights Issuer Service during this timeslot. If DeviceAddress and DeviceAddressMask fields are not present, this timeslot may be used to address all devices in the group. If the EuroCryptGroupType, EuroCryptGroup and EuroCryptGroupMask fields are not present, but DeviceAddress and DeviceAddressMask fields are, then the DeviceAddress and DeviceAddressMask fields are absolute device addresses using the longform_udn(), as defined in 9.3.2.6.

If the CertificateChainUpdate field is set to true, this rights issuer service will contain CertificateChainUpdates within this timeslot. These apply to all devices registered with the rights issuer using this rights issuer service. The use of certificate chain updates is described in 13.7.7.

EuroCryptGroup: The EuroCrypt Group address for the group(s) to be addressed in this scheduled time slot in a rights issuer service. A device can determine whether its EuroCrypt Group matches this value by bitwise ANDing its EuroCrypt Group address with the EuroCryptGroupMask (see below). If the result equals the value in this field, the groups match. The EuroCrypt Group address is the group_address, which is part of the unique_device_filter as defined in 9.3.2.7.2.

EuroCryptGroupMask: A mask to be applied to the EuroCryptGroup. See the definition for the EuroCryptGroup field above.

DeviceAddress: The device address for the device(s) to be addressed in this scheduled time slot in a rights issuer service. A device can determine whether its device address matches this value by bitwise ANDing its device address with the DeviceAddressMask (see below). If the result equals the value in this field, the device addresses match. In case the EuroCryptGroup and EuroCryptGroupMask fields are not present, the DeviceAddress is in the form of a longform_udn(), as defined in 9.3.2.6. In case the EuroCryptGroup and EuroCryptGroupMask fields are present, the DeviceAddress is the fixed_position_in_group, which is part of the unique_device_filter as defined in 9.3.2.7.2.

DeviceAddressMask: A mask to be applied to the DeviceAddress. See the definition for the DeviceAddress field above.

CertificateChainUpdate: If set to true, this timeslot will be used to broadcast certificate chain updates. Note that these apply to all devices registered with the rights issuer, not just those referenced by the above fields.

A.17.4 Event scheduling

This standard requires some extra permissions data to be carried related to event scheduling. The following structure is defined.

```
<complexType name="PermissionScheduleEventType">
    <complexContent>
        <extension base="esg:ScheduleEventType">
            <sequence>
                <element name="PermissionCategory" type="unsignedShort" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

A.17.5 Acquisition data

The key stream shall be signalled in the ESG acquisition data, as follows.

```
<complexType name="esg:KeyStreamType">
    <attribute name="IPDCKMSId" type="unsignedShort" use="required" />
    <attribute name="IPDCoperatorID" type="string" use="required" />
</complexType>
```

The IPDCoperatorID attribute is mapped to socID (when a single KSM stream is applicable to all COCs) or cocID (when there are COC-specific streams).

Furthermore, the following additional parameter to the SDP fmtp description shall be added.

Parameter	Mand/ Opt	Type	Comments
serviceBaseCID	M	string	Base ID of the service

When serviceBaseCID is used to construct a type that is a CID, for example service_CID or programme_CID in 7.2.4, and if serviceBaseCID starts with a prefix of the form “scheme:”, with ‘scheme’ as defined in IETF RFC 1738, for example cid:, the prefix shall be dropped when constructing the CID, since CIDs cannot contain more than one prefix.

A.17.6 XML schema definition

The schema definition below defines the namespace http://www.18crypt.com/2005/XMLSchema and all XML elements used in an ESG. The schema SHALL take precedence over XML elements declared previously in this clause.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:esg="urn:dvb:ipdc:esg:2005"
xmlns:mpeg7="urn:tva:mpeg7:2005" xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
xmlns:c18="http://www.18crypt.com/2005/XMLSchema"
targetNamespace="http://www.18crypt.com/2005/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
    <import namespace="urn:tva:mpeg7:2005" schemaLocation="../tva/3-1_CM_2005-06-
26/tva_mpeg7.xsd"/>
    <import namespace="urn:dvb:ipdc:esg:2005" schemaLocation="../DVB/ESG.xsd"/>
    <import namespace="urn:oma:bac:dldrm:roap-1.0" schemaLocation="../OMA/OMA-DRM-ROAP-
V2_0.xsd"/>
    <complexType name="ServiceOperationCentreType">
        <complexContent>
            <extension base="esg:PrivateDataType">
                <sequence>
                    <element name="socID" type="anyURI"/>
                    <element name="socName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
                    <element name="socKeyURL" type="anyURI"/>
                    <element name="socInfoURL" type="anyURI"/>
                    <element name="socRiProxyURL" type="anyURI" minOccurs="0"/>
                </sequence>
            </extension>
```

```
        </complexContent>
    </complexType>
    <complexType name="CustomerOperationCentreType">
        <complexContent>
            <extension base="esg:PrivateDataType">
                <sequence>
                    <element name="cocID" type="anyURI"/>
                    <element name="cocName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
                    <element name="cocLocalFlag" type="boolean"/>
                    <element name="cocRekeyingSafetyWindow" type="integer"/>
                    <element name="cocPurchaseURL" type="anyURI" minOccurs="0"/>
                    <element name="cocPortalURL" type="anyURI" minOccurs="0"/>
                    <element name="cocContactInfo" type="mpeg7:TextualType"/>
                    <element name="cocRiID" type="roap:Identifier"/>
                    <element name="cocRiURL" type="anyURI"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="HorizontalServiceInformationType">
        <complexContent>
            <extension base="esg:PrivateDataType">
                <sequence>
                    <element name="serviceBaseCID" type="string"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name="subscriptionTypeEnumType">
        <restriction base="integer">
            <enumeration value="1"/>
            <enumeration value="2"/>
            <enumeration value="3"/>
        </restriction>
    </simpleType>
    <complexType name="HorizontalPurchaseItemType">
        <complexContent>
            <extension base="esg:PurchaseDataBaseType">
                <sequence>
                    <element name="pItemName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
                    <element name="pItemVersion" type="integer"/>
                    <element name="purchaseOption">
                        <complexType>
                            <sequence>
                                <element name="SubscriptionUnit" nillable="true"
minOccurs="0">
                                    <complexType>
                                        <attribute name="subscriptionTypeEnum"
type="c18:subscriptionTypeEnum" use="required"/>
                                        <attribute name="subscriptionDuration" type="duration"
use="required"/>
                                    </complexType>
                                </element>
                                <element name="UnitText" type="mpeg7:TextualType"
minOccurs="0" maxOccurs="unbounded"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
                <attribute name="pItemId" type="unsignedInt" use="required"/>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name="subscriptionTypeEnum">
        <union memberTypes="c18:subscriptionTypeEnumType integer"/>
    </simpleType>
    <complexType name="RIServiceDataType">
        <complexContent>
            <extension base="esg:PrivateDataType">
                <sequence>
                    <element name="riID" type="roap:Identifier"/>
                    <element name="scheduled" type="boolean"/>
                    <element name="drmID" type="anyURI" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="RIAddressingType">
```

```
        <complexContent>
            <extension base="esg:PrivateDataType">
                <sequence>
                    <element name="EuroCryptGroup" type="unsignedLong"/>
                    <element name="EuroCryptGroupMask" type="unsignedLong"/>
                    <element name="DeviceAddress" type="unsignedByte" minOccurs="0"/>
                    <element name="DeviceAddressMask" type="unsignedByte" minOccurs="0"/>
                    <element name="CertificateChainUpdate" type="boolean"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="PermissionScheduleEventType">
        <complexContent>
            <extension base="esg:ScheduleEventType">
                <sequence>
                    <element name="PermissionCategory" type="unsignedShort" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

A.17.7 Mapping of required service guide data to ESG of IPDC over DVB-H systems

This subclause specifies the mapping of the required service guide data (12.5) to the ESG of IPDC over DVB-H systems (Table A.16).

Table A.16 – Mapping of required service guide data to the IPDC ESG

Required data		IPDC ESG data	
Field name	**Element**	**Field name**	**Element**
socID	SOC	socID (IP platform ID)	ServiceOperationCentreType via PrivateData element in the PurchaseChannel fragment
socName	SOC	socName	ServiceOperationCentreType via PrivateData element in the PurchaseChannel fragment
socKeyURL	SOC	socKeyURL	ServiceOperationCentreType via PrivateData element in the PurchaseChannel fragment
socInfoURL	SOC	socInfoURL	ServiceOperationCentreType via PrivateData element in the PurchaseChannel fragment
socRiProxyURL	SOC	socRiProxyURL	ServiceOperationCentreType via PrivateData element in the PurchaseChannel fragment
cocID	COC	cocID	CustomerOperationCentreType via PrivateData element in the PurchaseChannel fragment
cocLocalFlag	COC	cocLocalFlag	CustomerOperationCentreType via PrivateData element in the PurchaseChannel fragment
cocName	COC	cocName	CustomerOperationCentreType via PrivateData element in the PurchaseChannel fragment
cocRekeyingSafetyWindow	COC	cocRekeyingSafetyWindow	CustomerOperationCentreType via PrivateData element in the PurchaseChannel fragment
cocPurchaseURL	COC	cocPurchaseURL	CustomerOperationCentreType via PrivateData element in the PurchaseChannel fragment
cocPortalURL	COC	cocPortalURL	CustomerOperationCentreType via PrivateData element in the PurchaseChannel fragment
cocContactInfo	COC	cocContactInfo	CustomerOperationCentreType via PrivateData element in the PurchaseChannel fragment

Required data		IPDC ESG data	
Field name	**Element**	**Field name**	**Element**
cocRiID	COC	cocRiID	CustomerOperationCentreType via PrivateData element in the PurchaseChannel fragment
serviceID	Service	serviceID	ServiceType
serviceTypeEnum	Service	ServiceType (controlled term)	ServiceType
serviceName	Service	ServiceName	ServiceType
serviceDescription	Service	ServiceDescription	ServiceType
serviceBaseCID	Service	serviceBaseCID	HorizontalServiceInformationType via PrivateData field in ServiceType
sItemID	ScheduleItem	Not applicable	
sItemName	ScheduleItem	Not applicable (are available via content referencing)	
sItemDescription	ScheduleItem	Not Applicable (are available via content referencing)	
sItemStartTime	ScheduleItem	PublishedStartTime	ScheduleEventType
sItemEndTime	ScheduleItem	PublishedEndTime	ScheduleEventType
cItemID	ContentItem	contentID (anyURI)	ContentType
cItemTypeEnum	ContentItem	ContentType (controlled Term)	ContentType
cItemName	ContentItem	Title	ContentType
cItemDescription	ContentItem	Synopsis	ContentType
cItemGroupRange	ContentItem	Not Applicable	
cItemDeviceRange	ContentItem	Not Applicable	
cItemDomainRange	ContentItem	Not Applicable	
pItemID	Purchase Item	pItemId	HorizontalPurchaseItemType
pItemVersion	Purchase Item	pItemVersion	HorizontalPurchaseItemType
pItemName	Purchase Item	pItemName	HorizontalPurchaseItemType
pItemDescription	Purchase Item	pItemDescription	HorizontalPurchaseItemType
pItemServiceID	Purchase Item	ServiceRef, Referred to via ServiceBundleRef in Purchase Fragment	ServiceBundleType
pItemScheduleItemID	Purchase Item	No direct equivalent. Reference is via ServiceBundle which has a ContentType	
pItemContentItemID	Purchase Item	No direct equivalent. Reference is via ServiceBundle which has a ContentType	
cocID	Purchase Data	Linked to via PurchaseChannelIDRef	PurchaseRequestType
purchaseOption	Purchase Data	purchaseOption	HorizontalPurchaseItemType

A.18 Mapping of required service guide data to PSI/SI tables of DVB T/C/S systems

This clause specifies the mapping of the required service guide data (12.5) to the PSI/SI tables of DVB T/C/S systems (Table A.17).

Table A.17 – Mapping of required service guide data to DVB PSI/SI tables

Required data		DVB-T/C/S PSI/SI table data	
Field name	**Element**	**Field name**	**Table**
socID	SOC	original_network_id	NIT
socName	SOC	network_name_descriptor()	NIT
socKeyURL	SOC	key_url_descriptor()	NIT
socInfoURL	SOC	info_url_descriptor()	NIT
socRiProxyURL	SOC	Not needed (roaming only).	
cocID	COC	coc_id	PCT
cocLocalFlag	COC	Not needed (roaming only).	
cocName	COC	coc_name	PCT
cocRekeyingSafetyWindow	COC	coc_rekeying_safety_window	PCT
cocPurchaseURL	COC	coc_purchase_url	PCT
cocPortalURL	COC	coc_portal_url	PCT
cocContactInfo	COC	coc_contact_info, telephone_descriptor()	PCT
cocRiID	COC	Link via short_rights_issuer_id.	PCT
serviceID	Service	short_service_base_CID in service_ID_descriptor() or uniquely via TSID.ONID.ServiceID	SDT
serviceTypeEnum	Service	service_type in service_descriptor(), service data is not a service on its own in DVB-T/C/S. Newly defined RI_SERVICE_TYPE.	SDT
serviceName	Service	service_name in service_descriptor() or in multilingual_service_name_descriptor()	SDT
serviceDescription	Service	No equivalent; should be done via ServiceBundle.	
serviceBaseCID	Service	serviceBaseCID in service_ID_descriptor()	SDT
sItemID	ScheduleItem	event_id, unique together with service_id it links to.	EIT
sItemName	ScheduleItem	event_name in short_event_descriptor()	EIT
sItemDescription	ScheduleItem	short_event_descriptor() or extended_event_descriptor()	EIT
sItemStartTime	ScheduleItem	start_time	EIT
sItemEndTime	ScheduleItem	start_time + duration	EIT
cItemID	ContentItem	event_id, unique together with service_id it links to.	EIT
cItemTypeEnum	ContentItem	Not applicable	
cItemName	ContentItem	event_name in short_event_descriptor()	EIT
cItemDescription	ContentItem	short_event_descriptor() or extended_event_descriptor()	EIT
cItemGroupRange	ContentItem	Not applicable	

Required data		DVB-T/C/S PSI/SI table data	
Field name	**Element**	**Field name**	**Table**
cItemDeviceRange	ContentItem	Not applicable	
cItemDomainRange	ContentItem	Not applicable	
pItemID	Purchase Item	purchase_id in purchase_item_descriptor()	PIT/EIT/SDT
pItemVersion	Purchase Item	purchase_item_version in purchase_item_descriptor()	PIT/EIT/SDT
pItemName	Purchase Item	first multilingual_text() in purchase_item_descriptor()	PIT/EIT/SDT
pItemDescription	Purchase Item	second multilingual_text() in purchase_item_descriptor()	PIT/EIT/SDT
pItemServiceID	Purchase Item	short_service_base_CID in SBT or directly when purchase_item_descriptor is used in SDT/EIT.	SBT/SDT/EIT
pItemScheduleItemID	Purchase Item	event_id in SBT or directly when purchase_item_descriptor is used in EIT.	SBT/EIT
pItemContentItemID	Purchase Item	No equivalent as Schedule and Content are both handled by EIT.	
cocID	Purchase Data	Link via coc_channel_id in purchase_item_descriptor().	PIT/SDT/EIT
purchaseOption	Purchase Data	purchase_item_descriptor()	PIT/SDT/EIT

Figure A.9 shows the relationships between the tables in Table A.17.

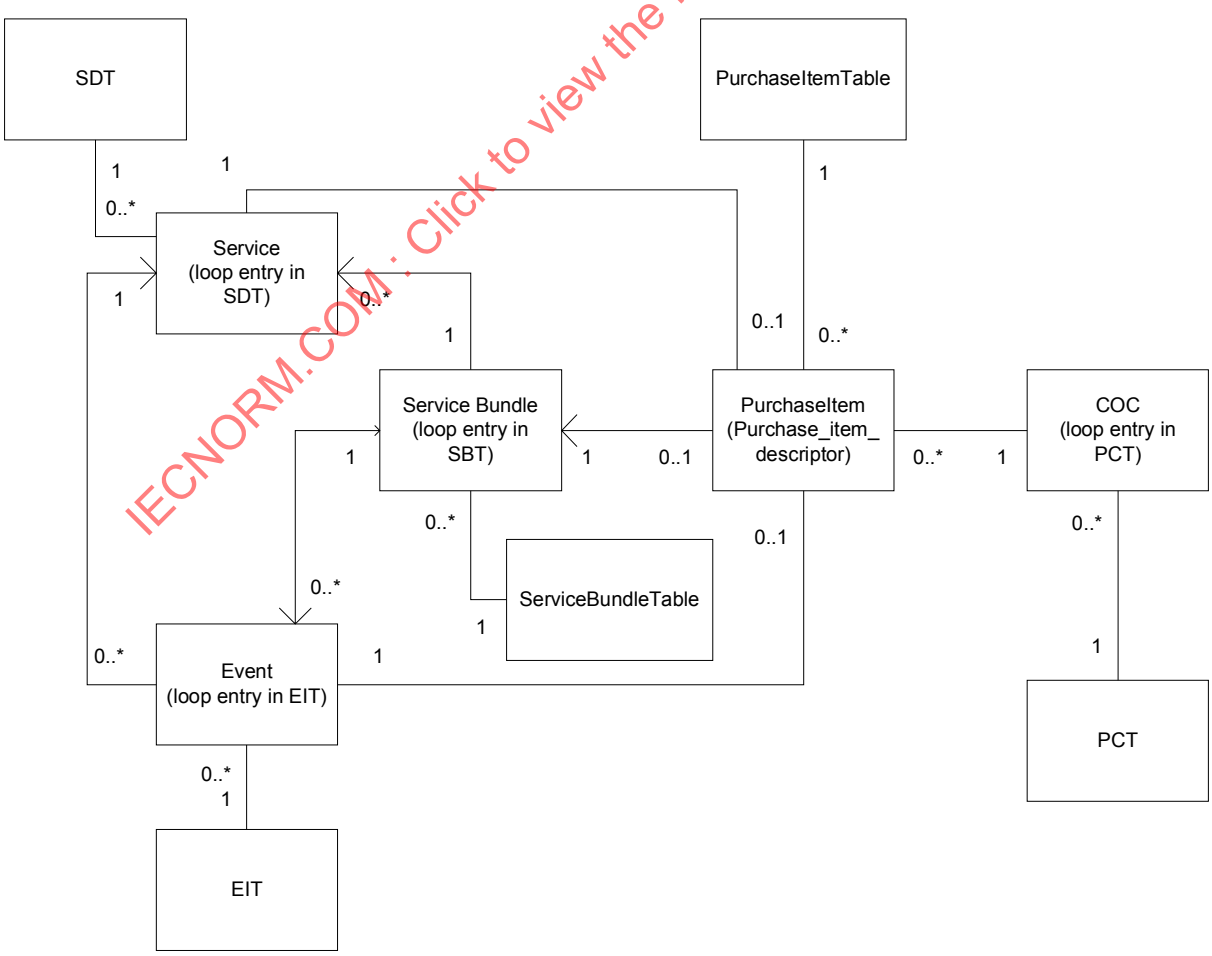


Figure A.9 – Relationship between DVB-T/C/S PSI/SI tables

A.19 Service guide for TV anytime based systems

A.19.1 General

This clause specifies some generic extensions to the TV anytime metadata schema in ETSI TS 102 822-3-1 to support this standard.

In TV anytime systems, the PurchaseItem fragment carries purchase information, and can be associated with any of the following fragments.

- ProgramInformation fragment, describing an individual programme.
- GroupInformation fragment, which associates a number of programmes into a logical group, for example, a series or collection.
- Schedule fragment describes a specific scheduled event.
- ServiceInformation fragment describes a service – note that much of this information can also be carried in the SD&S data.
- OnDemandProgram fragment describes a content item that is available "on demand".

Any association allowed by the TV anytime specification may be used.

This standard adds the following types to TV anytime.

- PurchaseChannelType – new type used to provide information on how to purchase content from a particular customer operations centre.
- ServiceBundleType – new type used to describe a bundle of services which may be purchased together.
- UniversalPurchaseItemType – extension of PurchaseItemType with additional information required by this standard.
- UniversalServiceInformationType – extension of ServiceInformationType with additional information required by this standard.
- UniversalOnDemandServiceType – extension of OnDemandServiceType with additional information required by this standard.

UniversalPurchaseType – extension of PurchaseType with additional information required by this standard.

A.19.2 Relationships between types

Figure A.10 shows the relationships between the various types. Only relationships relevant to this standard are shown.

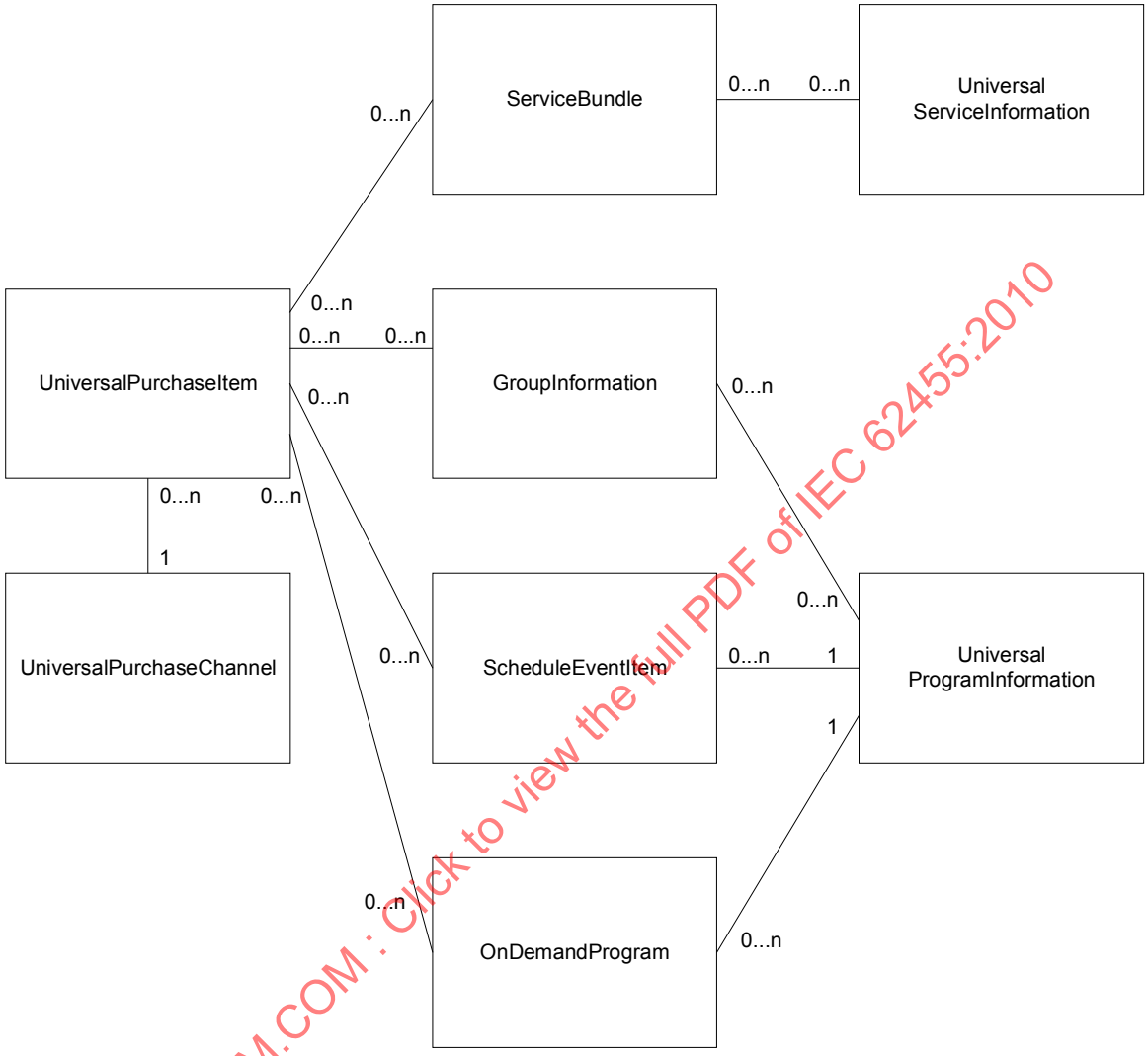


Figure A.10 – Relationships between the defined types

A.19.3 Mapping of required service guide data to TV anytime

This subclause specifies the mapping of the required service guide data (12.5) to the TV-anytime metadata (Table A.18).

Table A.18 – Mapping of required service guide data to IPI BCG/TV anytime

Required data		IPI BCG/TV anytime	
Field name	**Element**	**Field name**	**Element**
socID	SOC	socID	SD&S and/or UniversalServiceInformationType
socName	SOC	socName	PurchaseChannel
socKeyURL	SOC	socKeyURL	PurchaseChannel
socInfoURL	SOC	socInfoURL	PurchaseChannel
socRiProxyURL	SOC	socRiProxyURL[a]	PurchaseChannel

Required data		IPI BCG/TV anytime	
Field name	**Element**	**Field name**	**Element**
cocID	COC	cocID	PurchaseChannel
cocLocalFlag	COC	cocLocalFlag[a]	PurchaseChannel
cocName	COC	cocName	PurchaseChannel
cocRekeyingSafetyWindow	COC	cocRekeyingSafetyWindow	PurchaseChannel
cocPurchaseURL	COC	cocPurchaseURL	PurchaseChannel
cocPortalURL	COC	cocPortalURL	PurchaseChannel
cocContactInfo	COC	cocContactInfo	PurchaseChannel
cocRiID	COC	cocRiID	PurchaseChannel
serviceID	Service	serviceID	ServiceInformationType or SD&S
serviceTypeEnum	Service	Signalling of RI service is not applicable. Signalling of other service types is beyond the scope of this standard	
serviceName	Service	Name	ServiceInformationType
serviceDescription	Service	ServiceDescription	ServiceInformationType
serviceBaseCID	Service	serviceBaseCID	SD&S and UniversalServiceInformationType
sItemID	ScheduleItem	Not applicable	
sItemName	ScheduleItem	Name from ProgramInformation	ScheduleEventType
sItemDescription	ScheduleItem	Description from ProgramInformation	ScheduleEventType
sItemStartTime	ScheduleItem	PublishedStartTime	ScheduleEventType
sItemEndTime	ScheduleItem	PublishedEndTime	ScheduleEventType
cItemID	ContentItem	Not applicable	
cItemTypeEnum	ContentItem	Not applicable	
cItemName	ContentItem	Title	BasicContentDescriptionType
cItemDescription	ContentItem	Synopsis	BasicContentDescriptionType
cItemGroupRange	ContentItem	Not applicable	
cItemDeviceRange	ContentItem	Not applicable	
cItemDomainRange	ContentItem	Not applicable	
pItemID	Purchase Item	pItemId	UniversalPurchaseItemType
pItemVersion	Purchase Item	pItemVersion	UniversalPurchaseeItemType
pItemName	Purchase Item	pItemName	UniversalPurchaseeItemType
pItemDescription	Purchase Item	Description	PurchaseItemType
pItemServiceID	Purchase Item	ServiceInformation fragment associated with the PurchaseItem	

Required data		IPI BCG/TV anytime	
Field name	**Element**	**Field name**	**Element**
pItemScheduleItemID	Purchase Item	ScheduleItem fragment associated with the PurchaseItem	
pItemContentItemID	Purchase Item	ProgramInformation fragment associated with the PurchaseItem	
cocID	Purchase Data	Linked via PurchaseChannelIDRef.	PurchaseRequestType
purchaseOption	Purchase Data	Various	UniversalPurchaseItemType

[a] These entries are retained for consistency with IPDC over DVB-H systems, but their use is not defined by this standard.

A.19.4 XML types

A.19.4.1 Namespace

The namespace used for the following XML elements is xmlns:iptv="http://www.18crypt.com/2005/IPTV/XMLSchema"

A.19.4.2 SOC identifier

Figure A.11 shows the XML fragment used to carry the SOC identifier.

```
<xs:simpleType name="soc_ID">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
```

Figure A.11 – XML fragment for SOC identifier

A.19.4.3 ServiceBaseCID

Figure A.12 shows the XML fragment used to carry the serviceBaseCID.

```
<xs:simpleType name="serviceBaseCID">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
```

Figure A.12 – XML fragment for serviceBaseCID

When serviceBaseCID is used to construct a type that is a CID, e.g. service_CID or programme_CID in 7.2.4, and if serviceBaseCID starts with a prefix of the form scheme:", with 'scheme' as defined in IETF RFC 1738, e.g. cid:, the prefix shall be dropped when constructing the CID, since CIDs cannot contain more than one prefix.

A.19.4.4 UniversalPurchaseItemType

Figure A.13 shows the XML type UniversalPurchaseItemType. This extends PurchaseItemType, as defined in ETSI TS 102 822-3-1.

```
<complexType name="UniversalPurchaseItemType">
    <complexContent>
        <extension base="tva:PurchaseItemType">
            <sequence>
                <element name="pItemName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
                <element name="pItemVersion" type="integer"/>
                <element name="purchaseOption">
                    <complexType>
                        <sequence>
                            <element name="SubscriptionUnit" nillable="true" minOccurs="0">
                                <complexType>
                                    <attribute name="c18:subscriptionTypeEnum"
type="esg:subscriptionTypeEnum" use="required"/>
                                    <attribute name="subscriptionDuration" type="duration"
use="required"/>
                                </complexType>
                            </element>
                            <element name="UnitText" type="mpeg7:TextualType" minOccurs="0"
maxOccurs="unbounded"/>
                        </sequence>
                    </complexType>
                </element>
            </sequence>
            <attribute name="pItemId" type="unsignedInt" use="required"/>
        </extension>
    </complexContent>
</complexType>
<simpleType name="subscriptionTypeEnumType">
    <restriction base="integer">
        <enumeration value="1"/>
        <enumeration value="2"/>
        <enumeration value="3"/>
    </restriction>
</simpleType>
<simpleType name="subscriptionTypeEnum">
    <union memberTypes="esg:subscriptionTypeEnumType integer"/>
</simpleType>
```

NOTE The item description is part is the PurchaseItemType and so is not separately carried in the UniversalPurchaseItemType.

Figure A.13 – Definition of UniversalPurchaseItemType

Other fields are as defined for IPDC over DVB-H systems in A.17.2.5.

A.19.4.5 ServiceBundleType

Figure A.14 defines the ServiceBundleType.

```
<complexType name="ServiceBundleType">
    <sequence>
        <element name="ServiceBundleName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
        <element name="ServiceBundleProvider" type="string" minOccurs="0"/>
        <element name="ServiceBundleMediaTitle" type="mpeg7:TitleMediaType" minOccurs="0"/>
        <element name="ServiceBundleDescription" type="mpeg7:TextualType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="ServiceBundleGenre" type="tva:GenreType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="ServiceRef" type="tva:TVAIDRefType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="ParentalGuidance" type="mpeg7:ParentalGuidanceType" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="serviceBundleID" type="anyURI" use="required"/>
</complexType>
```

Figure A.14 – Definition of the ServiceBundleType

ServiceRef refers to the serviceRefID field in the ServiceInformation fragment of each referenced service.

The use of all fields is as specified in ETSI TS 102 471.

A.19.4.6 PurchaseChannelType

The PurchaseChannelType definition is as defined in ETSI TS 102 471 for IPDC over DVB-H systems. The PrivateData field shall contain the following types.

- ServiceOperationCentreType, as defined in Clause A.17.
- CustomerOperationCentreType, as defined in Clause A.17.

A.19.4.7 UniversalServiceInformationType

Figure A.15 defines the UniversalServiceInformationType.

```
<complexType name="UniversalServiceInformationType">
    <complexContent>
        <extension base="tva:ServiceInformationType">
            <sequence>
                <element name="socID" type="iptv:soc_ID"/>
                <element name="serviceBaseCID" type="xs:string"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure A.15 – Definition of UniversalServiceInformationType

Where the socID and/or serviceBaseCID are also carried in the broadcast discovery record in SD&S data, it should match the value provided here. In the event that the values differ, the precedence shall be as defined in ETSI TS 102 539.

A.19.4.8 UniversalOnDemandServiceType

Figure A.16 defines the UniversalOnDemandServiceType.

```
<complexType name="UniversalOnsDemandServiceType">
    <complexContent>
        <extension base="tva:OnDemandServiceType">
            <sequence>
                <element name="socID" type="iptv:soc_ID"/>
                <element name="serviceBaseCID" type="xs:string"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure A.16 – Definition of UniversalOnDemandServiceType

Where the socID and/or serviceBaseCID are also carried in the content on demand record in SD&S data, it should match the value provided here. In the event that the values differ, the precedence shall be as defined in ETSI TS 102 539.

A.19.4.9 UniversalPurchaseType

Figure A.17 defines the UniversalPurchaseType. SubscriptionTypeEnum values are as follows.

- 1 for continuous subscription
- 2 for one-time subscription
- 3 for preview

```
<complexType name="UniversalPurchaseType">
    <complexContent>
        <extension base="tva:PurchaseType">
            <sequence>
                <element name="PurchaseRequest" type="iptv:PurchaseRequestType" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="PurchaseRequestType">
    <sequence>
        <element name="DRMSystem" type="anyURI"/>
        <element name="PurchaseData" type="UniversalPurchaseItemType" minOccurs="0"/>
        <element name="PurchaseChannelIDRef" type="ESGIDRefType" minOccurs="0"/>
    </sequence>
</complexType>
```

Figure A.17 – Definition of UniversalPurchaseType

The definition of DRMSystem is the same as for CA_System_ID, see 15.7.2.4.1.

A.19.4.10 XML schema definition

The schema definition below defines the namespace xmlns:iptv="http://www.18crypt.com/2005/IPTV/XMLSchema" and all XML elements used. The schema SHALL take precedence over XML elements declared previously in this clause.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:esg="urn:dvb:ipdc:esg:2005"
xmlns:mpeg7="urn:tva:mpeg7:2005" xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
xmlns:tva="urn:tva:metadata:2005" xmlns:c18="http://www.18crypt.com/2005/XMLSchema"
xmlns:iptv="http://www.18crypt.com/2005/IPTV/XMLSchema"
targetNamespace="http://www.18crypt.com/2005/IPTV/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
    <import namespace="urn:tva:mpeg7:2005" schemaLocation="../tva/3-1_CM_2005-06-
26/tva_mpeg7.xsd"/>
    <import namespace="urn:dvb:ipdc:esg:2005" schemaLocation="../DVB/ESG.xsd"/>
    <import namespace="urn:tva:metadata:2005" schemaLocation="../tva/tva_metadata_3-
1_v131.xsd"/>
    <import namespace="urn:oma:bac:dldrm:roap-1.0" schemaLocation="../OMA/OMA-DRM-ROAP-
V2_0.xsd"/>
    <import namespace="http://www.18crypt.com/2005/XMLSchema"
schemaLocation="../IEC/18crypt_esg.xsd"/>
    <complexType name="CA_ID_descriptor">
        <sequence>
            <element name="CA_System_ID" type="anyURI"/>
        </sequence>
    </complexType>
    <simpleType name="soc_ID">
        <restriction base="string"/>
    </simpleType>
    <simpleType name="serviceBaseCID">
        <restriction base="string"/>
    </simpleType>
    <complexType name="UniversalPurchaseItemType">
        <complexContent>
            <extension base="tva:PurchaseItemType">
                <sequence>
                    <element name="pItemName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
                    <element name="pItemVersion" type="integer"/>
                    <element name="purchaseOption">
                        <complexType>
                            <sequence>
                                <element name="SubscriptionUnit" nillable="true"
minOccurs="0">
                                    <complexType>
                                        <attribute name="subscriptionTypeEnum"
type="c18:subscriptionTypeEnum" use="required"/>
                                        <attribute name="subscriptionDuration" type="duration"
use="required"/>
                                    </complexType>
                                </element>
```

```
                                <element name="UnitText" type="mpeg7:TextualType"
minOccurs="0" maxOccurs="unbounded"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
                <attribute name="pItemId" type="unsignedInt" use="required"/>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="ServiceBundleType">
    <sequence>
        <element name="ServiceBundleName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
        <element name="ServiceBundleProvider" type="string" minOccurs="0"/>
        <element name="ServiceBundleMediaTitle" type="mpeg7:TitleMediaType" minOccurs="0"/>
        <element name="ServiceBundleDescription" type="mpeg7:TextualType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="ServiceBundleGenre" type="tva:GenreType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="ServiceRef" type="tva:TVAIDRefType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="ParentalGuidance" type="mpeg7:ParentalGuidanceType" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="serviceBundleID" type="anyURI" use="required"/>
</complexType>
<complexType name="UniversalServiceInformationType">
    <complexContent>
        <extension base="tva:ServiceInformationType">
            <sequence>
                <element name="socID" type="iptv:soc_ID"/>
                <element name="serviceBaseCID" type="string"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="UniversalOnsDemandServiceType">
    <complexContent>
        <extension base="tva:OnDemandServiceType">
            <sequence>
                <element name="socID" type="iptv:soc_ID"/>
                <element name="serviceBaseCID" type="string"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="UniversalPurchaseType">
    <complexContent>
        <extension base="tva:PurchaseType">
            <sequence>
                <element name="PurchaseRequest" type="iptv:PurchaseRequestType" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="PurchaseRequestType">
    <sequence>
        <element name="DRMSystem" type="anyURI"/>
        <element name="PurchaseData" type="iptv:UniversalPurchaseItemType" minOccurs="0"/>
        <element name="PurchaseChannelIDRef" type="esg:ESGIDRefType" minOccurs="0"/>
    </sequence>
</complexType>

</schema>
```

A.20 The use and precedence of programme ROs, service ROs, the permissions_category and access criteria

A.20.1 General

An operator can give a user access to a service using a service RO. The service RO is typically valid for a relatively long period, for example, a month. A service RO contains permissions and constraints for all programmes in the service.

In the simplest case, the service RO contains one set of permissions and constraints, which means that all programmes in the service have the same permissions and constraints for users that have received such a service RO.

An operator might wish to treat certain programmes of a service in a special way. For example, the operator may wish to sell additional permissions to users for specific programmes or give different restrictions to certain programmes for users who just have the subscription (for example, more restrictions for a new movie, less restrictions for preview type content). This standard provides two mechanisms to achieve such a requirement.

A.20.2 Use of multiple programme ROs in addition to a service RO

The first mechanism this standard provides for having different permissions and constraints for programmes is to have a programme RO for each programme for which the permissions and constraints are different from the ones in the encompassing service RO. Therefore, a programme RO of a programme shall have precedence over a service RO of the service for which the programme is a part.

These programme ROs can be broadcast to broadcast devices and they can be sent over an interactivity channel to devices that have an interactivity channel. The distribution of these programme ROs can consume much bandwidth in case there are many programmes for which the permissions and constraints are different from the service RO.

A.20.3 Use of the permissions_category and service ROs

The second mechanism this standard provides for having different permissions and constraints for programmes is by using the permissions_category functionality (see specification of permissions_category). In this second mechanism, only service ROs are distributed, which can save considerable bandwidth in comparison with the first mechanism, where multiple programme ROs are distributed.

For BCROs, it is possible to have more than one set of permissions and constraints for each asset in the service RO. Each of these sets is identified by an 8-bit number that is called the permissions_category. For service ICROs, the permissions_category is indicated in their Service_CID (see specification of permissions_category). The key stream message can contain a permissions_category number. This number in the key stream message indicates which of the sets of permissions and constraints in the service BCRO and which service ICRO applies to the programme that is broadcast at the time of reception of the permissions_category number in the key stream message. The permissions_category should remain constant over one programme.

Using the permissions_category number functionality, it is possible to have a service with programmes with different permissions and constraints by just distributing a service RO containing multiple sets of permissions and constraints, each set indexed by permissions_category.

The RO as defined in OMA DRM 2.0 does not have the permissions_category functionality but it can contain different rights for different assets. Taking advantage of this feature, this standard contains an enhancement that defines this functionality (see the specification of the permissions_category field).

A.20.4 Use of programme ROs without a service RO

An operator can give access to the user on a programme-by-programme basis using programme ROs, without using a service RO. This may be useful in, for example, pay-per-view business models.

A.20.5 Access criteria

The key stream message may contain access criteria in the form of access_criteria_descriptors; see 7.2.2. This standard specifies three access criteria, as follows:

- parental rating, see 7.2.2.2;
- copy control information, see 7.2.2.3; and
- black-out spot-beam, see 7.2.2.4.

For parental rating, this standard does not offer other equivalents to transfer parental rating information to the device. This means that the precedence as specified in A.20.6 is not applicable to parental rating. In particular, programme ROs cannot override a denial of access by parental rating.

Copy control information, however, can also be transferred to the device in rights objects or in an access_criteria_descriptor in the key stream, see Clause A.21. Therefore, the precedence as defined in A.20.6 also applies to the different ways the copy control information can be sent to the device.

If a blackout – spotbeam access_criteria_descriptor is present in the KSM, the outcome of the blackout – spotbeam mechanism as specified in 7.2.2.4.3 is used, in the precedence rules as defined in A.20.6.

A.20.6 Precedence of permissions and constraints in programme and service ROs and access criteria

The precedence of permissions and constraints in programme and service ROs and access criteria are specified below from highest precedence to lowest.

a) The permissions and constraints in a programme RO for a programme in a service shall have precedence over any set of permissions and constraints in any service RO applicable to that service. This is independent of whether or not a set of permissions and constraints is indicated for that programme in the key stream message by the permissions_category field. Access criteria related permissions and constraints in a programme RO for a programme in a service shall also have precedence over access criteria in the key stream message for that programme.

b) In the absence of a programme RO for a programme in a service, the access criteria in the key stream message for that programme shall have precedence over any other set of ‘access criteria related permissions and constraints’ (see A.20.5) in any service RO that is applicable to the service of which that programme is a part. This is independent of whether or not a set of permissions and constraints is indicated for that programme in the key stream message by the permissions_category field.

c) Furthermore in the absence of a programme RO for a programme in a service, the set of permissions and constraints that is indicated for that programme in the key stream message by the permissions_category field shall have precedence over any other set of permissions and constraints in any service RO that is applicable to the service of which that programme is a part.

d) If none of the higher precedence items defined in the above three rules are available, all permissions and constraints in service ROs that are applicable to the service of which the programme is a part, and that are specified for a value of 0 for permissions_category in Table 26 shall apply.

The black-out-spotbeam mechanism is a special case. If the black-out-spotbeam mechanism yields "bosb_access_denied" (see 7.2.2.4.3) for a device, access based on service ROs and/or permissions_category fields in the key stream message is always denied. However, programme ROs override any access denials of the black-out-spotbeam mechanism. If the black-out-spotbeam mechanism yields "bosb_access_allowed" (see 7.2.2.4.3), a device still needs a proper RO for being able to access the programme or service.

A.21 Copy control information in KSM and ROs

Copy control information controls analogue and digital outputs and the creation of copies usable by other protection systems. By "creation of copies" is meant the creation of a (digital) representation of the content and possibly the creation of other information (for example, an RO), such that other devices using other protection systems can render the content. Simply copying of encrypted content is not meant with "creation of copies", since other devices cannot use that content without proper ROs. Also, the copying of content for which a device has a domain RO for other devices in the same domain is not meant with "creation of copies", and neither is the creation of superdistributable recordings, which is controlled by the save permission. Thus, copy control information does not prevent the creation of copies that are governed by OMA DRM as permitted by the relevant service or programme ROs. Therefore, copy control information is important when exporting content to other, possibly analogue, systems.

Copy control information for use in the key stream message is specified in 7.2.2.3. Copy control information can also be present in ROs. In the latter case, it is in the form of a parameter to the SYSTEM constraint[1] of an EXPORT permission. An example for an ICRO is:

```
<o-ex:permission>
   <o-dd:display/>
   <o-dd:print/>
   <oma-dd:export oma-dd:mode="copy">
    <o-ex:constraint>
     <oma-dd:system>
      <o-ex:context>
       <o-dd:version>1.0</o-dd:version>
       <o-dd:uid>urn:oma:drms:org-XYZ:System_XYZ?CCI_parameter=value_of_the_CCI</o-dd:uid>
      </o-ex:context>
     </oma-dd:system>
    </o-ex:constraint>
   </oma-dd:export>
</o-ex:permission>
```

As a special case, the value '0xFF' of the permissions_category in the key stream signals that no post-acquisition content protection is required, which means only the service is protected (see the specification of the permissions_category field in 7.2).

The specification of the copy control information parameter for each system, its effect in each system and the systems themselves are not within the scope of this standard and are left to the owners of those systems.

A.22 Recording, sharing and output of broadcast content

A.22.1 General

Broadcast content that is protected with the functionality offered by this standard is always encrypted on the broadcast channel.

What a device is allowed to do with the broadcast content is governed by

[1] The registration of system names for use in the SYSTEM constraint is managed by the Open Mobile Naming Authority (OMNA) of the Open Mobile Alliance.

- rights objects;
- permissions_category fields in the key stream message;
- CCI fields in the key stream message,

the precedence of which is specified in Clause A.20.

This clause specifies how recording, sharing and output of broadcast is controlled using the above mechanisms.

A.22.2 Service protection – Content protection

This standard may be used for service protection (see also 4.5). Service protection does not restrict the consumption of content after reception. If any such restrictions are needed, content protection may be used instead of, or in addition to, service protection.

Service protection (without any supplemental content protection) can be signalled in the following ways.

- The consumption of service or programme is governed by an RO that allows everything, i.e. it has ACCESS, PLAY and EXPORT permissions without any constraints.
- The consumption of service or programme is governed by an RO that has an EXPORT permission to an 'unencrypted' system, for example, the "urn:oma:drms:oma-drms:cleartext:1.0" system from OMA.
- The permissions_category field in the key stream message contains '0xFF' (see permissions_category specification).
- The CCI bits in the key stream or in an EXPORT permission indicate 'copying not restricted'; see 7.2.2.3.2.

In all other cases, the broadcast content is content protected.

Devices are restricted in the consumption of the content protected broadcast content. Devices shall protect content protected content in such a way that not-allowed consumption and access is not possible.

A.22.3 Recording of broadcast content

In the case of service protection (see A.22.2) recording of broadcast content is allowed in any format and may be done unencrypted.

In the case of content protection (see A.22.2) the following is allowed with respect to recording.

- Recordings shall be made in encrypted formats only; the encryption method shall be of equivalent strength or better than the encryption algorithm used for the broadcast itself; decryption keys shall be protected by the device and stored in secure memory. Playback of these encrypted recordings is allowed:
  - by the device itself which did the recording and only if it has an RO for the recorded broadcast with a PLAY permission;
  - by devices that belong to the same domain as the device that did the recording and then only if they have a domain RO for the recorded broadcast with the PLAY permission.
- Recordings in super-distributable formats shall not be made by the device, unless it has an RO with the SAVE permission in it; see Clause A.24. Playback of super-distributable recordings is only allowed by devices that have an RO with the PLAY permission for this recording.

- A device that implements this standard shall only allow devices or a subsystem implementing other standards to record broadcast content if has ROs with the proper EXPORT permissions and/or when the CCI bit settings allow recording using the other standard. The parameters to these EXPORT permissions and/or the CCI bit settings also determine what is allowed to be done with the recordings made using the other standard.

A.22.4 Sharing of broadcast content

A.22.4.1 General

This standard supports the (protected) sharing of broadcast content in the following ways.

A.22.4.2 Sharing of broadcast using domains

Broadcast content may be bound to domain ROs. This means that all devices belonging to the same domain can consume that content according to the permissions and constraints in the domain RO.

Examples:

- If a service or programme RO bound to a domain has the ACCESS permission, then all devices belonging to that domain can directly render the broadcast content during reception.
- If a service or programme RO bound to a domain has the PLAY permission, then all devices belonging to that domain can render recordings of content controlled by that service or programme RO, which are made by other devices belonging to the same domain. The recordings should be made in DCF or PDCF format and may be according to the super-distributable way, specified for the SAVE permission (Clause A.24).

A.22.4.3 Sharing of content using link level protection

ROs may contain EXPORT permissions using link-level protected systems (see NOTE 1). In compliance rules (see NOTE 2), the use of certain link-level protection systems may be granted without having appropriate ROs, or without the presence of equivalent CCI-bit settings in the key stream message.

NOTE 1 The target system of an EXPORT permission is indicated with a system name. The registration of system names is managed by the Open Mobile Naming Authority (OMNA) of the Open Mobile Alliance.

NOTE 2 The implementation of a device according to this International Standard may involve the signing of contracts with a trust authority. One of these contracts may be a 'compliance rule' contract, which may specify, for example, allowed outputs and content formats, disallowed outputs and formats, etc.

In the above cases, broadcast content may be streamed over the allowed protected links, while adhering to possible constraints or parameters in the EXPORT permission (see, for example, Clause A.21) and what is specified in compliance rules (see NOTE 2, above).

A.22.4.4 Sharing of content using EXPORT to other protected systems (in-home networking)

ROs may contain EXPORT permissions to other copy-controlled systems, or other DRM systems (see NOTE 1, above). Key stream messages may contain CCI-bit settings that allow the export to other copy-controlled systems.

The target systems of the EXPORT permission may be in-home network systems, in which the content can be freely shared between devices belonging to the in-home network, but not with devices that do not belong to the in-home network.

The target systems of the EXPORT permission may be systems in which a copy of the content is made on a physical medium which can be rendered by any device complying with the

specification of that system, but of which copy on that physical medium, no more copies can be made.

The target system, of the EXPORT permission may be any other protected or un-protected system.

In case of service protection, broadcast content can be freely shared after reception.

A.22.5 Output of broadcast content

In the case of service protection, any output is allowed.

In the case of content protection, the device, in outputting the content, shall adhere to what is allowed by the EXPORT permissions in ROs and CCI bit-settings in the key stream message.

For their analogue outputs, devices shall obey the CCI bits controlling analogue outputs; see 7.2.2.3.

For their digital outputs, devices shall obey the CCI bits controlling digital outputs; see 7.2.2.3. Furthermore for digital outputs, devices shall obey what is allowed with EXPORT permissions in ROs.

Compliance rules also may control the use of analogue and digital outputs.

A.23 Access permission

A.23.1 General

This clause specifies an extension to the OMA DRM REL V2.0 specification (see OMA-TS-DRM-REL-V2_0) called access permission that shall be used in an ICRO when granting a device the right to access (i.e. directly render) a service or programme. The access permission for use in BCROs is specified in 8.4.5.

The contents of this clause may be added to the OMA BCAST DLDRM specification (see OMA-TS-DRM-REL-V2_0). If this happens, that specification shall take precedence over this clause.

A.23.2 Specification of the access permission

The permission element from OMA-TS-DRM-REL-V2_0 is changed as shown in Table A.19.

Table A.19 – Updated permission element

Element	<!ELEMENT o-ex:permission (o-ex:constraint?, o-ex:asset*, o-dd:play?, o-dd:display?, o-dd:execute?, o-dd:print?, oma-dd:export?, oma-dd:access?, oma-dd:save?)>
Semantics	The <permission> element contains an optional <constraint> element, zero or more <asset> elements and a set of optional permissions specifying the rights over a piece of Content, such as <play>, <display>, <execute>, <print>, <export>, and <access> permission elements. The <constraint> element is the top-level constraint. As a sibling element to other permission elements such as <play>, <display> it applies to all sibling permission elements inside the same <permission> element. The DRM agent shall honour the top-level constraint in addition to honouring possible constraints specified as a child element to a permission element, for example, <play>, when granting access to content according to such a permission. The

	<asset> elements specified within the <permission> element enable expression linking allowing its sibling permission elements in the same <permission> element to apply to DRM content referenced by <asset> elements contained in an <agreement> element (i.e., outside a <permission> element). The link is established through the use of the "id" and "idref" attributes specified in OMA DRM (see OMA-TS-DRM-REL-V2_0). Note that the DRM agent shall respect both constraints specified as child elements to a permission element and those specified as top-level constraints in the same rights object, i.e., the stricter of two constraints of the same type prevails for a given permission element. Of course, rights objects with contradictory constraints should not be issued in the first place. When there is a top-level constraint that is otherwise not allowed as a child constraint to a permission, for example, <count> and <export mode="move">, the child constraint takes precedence over the top-level constraint as applied to this permission. For example, in the move scenario, content and rights object would be moved, and the <count> constraint would accordingly be removed, too. A DRM agent shall grant access to DRM content referenced by an <asset> element in the agreement model according to permissions specified inside a <permission> element that is as sibling elements to an <asset> element in the permission model, where the <asset> element referencing the DRM content and the <asset> element inside the <permission> element are linked by matching "id" and "idref" attributes. If no <asset> element is present in a permission element such as <play>, then the permission applies to all <asset> sibling elements in the same rights object. The <export> permission is associated with all of the DRM content referenced by <asset> elements within the same rights object. A single rights object has at most one <export> element within a given <permission> element.

NOTE The save permission from Clause A.24 is included in this table. However, this was not part of the accepted extension proposal for OMA DRM with access permission, because the one for the save permission came later. It is included here because it is required here after the acceptance of the save permission.

The access element is defined as shown in Table A.20.

Table A.20 – Access element

Element	<!ELEMENT oma-dd:access (o-ex:constraint)>
Semantics	The <access> element grants the permission to create an immediate rendering of audio or video content direct from a broadcast, multicast, or unicast stream during its reception. "Immediate" here means a time period in the order of one or a few seconds at most. It contains an optional <constraint> element. If the <constraint> element is specified the DRM agent shall grant access rights according to the <constraint> child element. If no <constraint> element is specified, the DRM agent shall grant unlimited access rights. A <system> element contained in a <constraint> child element to <access> is used to specify target system that may be used for creating an immediate rendering of the broadcast, multicast, or unicast stream during its reception. The <access> element has the semantics of rendering immediately the broadcast, multicast, or unicast stream somehow into user perceivable form. The DRM agent shall not grant access according to <access> to content that cannot be rendered in this way. Note that the DRM agent shall not grant access to stored content, not even stored broadcast, multicast, or unicast streams, based on the <access> permission. In order to specify rights for stored content, the <play> element shall be utilized instead.

A.24 Save permission

A.24.1 General

This clause specifies an extension to the OMA DRM REL V2.0 specification (see OMA-TS-DRM-REL-V2_0) called save permission that shall be used to allow the expression of a "recording" permission in interactivity channel and broadcast rights objects. It also specifies the procedure for recording broadcast content in OMA assets that are super-distributable and for which rights can be acquired using standard OMA DRM 2.0 procedures. The save permission for use in BCROs is specified in 8.4.4.

The contents of this clause may be added to the OMA BCAST DLDRM specification; see OMA-TS-DRM-REL-V2_0. If this happens, that specification shall take precedence over this clause.

A.24.2 Recording concept

The concept of controlled recording is illustrated in the Figure A.18. A rights issuer has issued a rights object to device A. This gives device A the right to access a certain broadcast asset, as well as the right to create a super-distributable copy of (part) of that broadcast asset in a new asset. Another device B may receive a copy of this new content file and contacts the rights issuer to acquire (play) rights for this content.

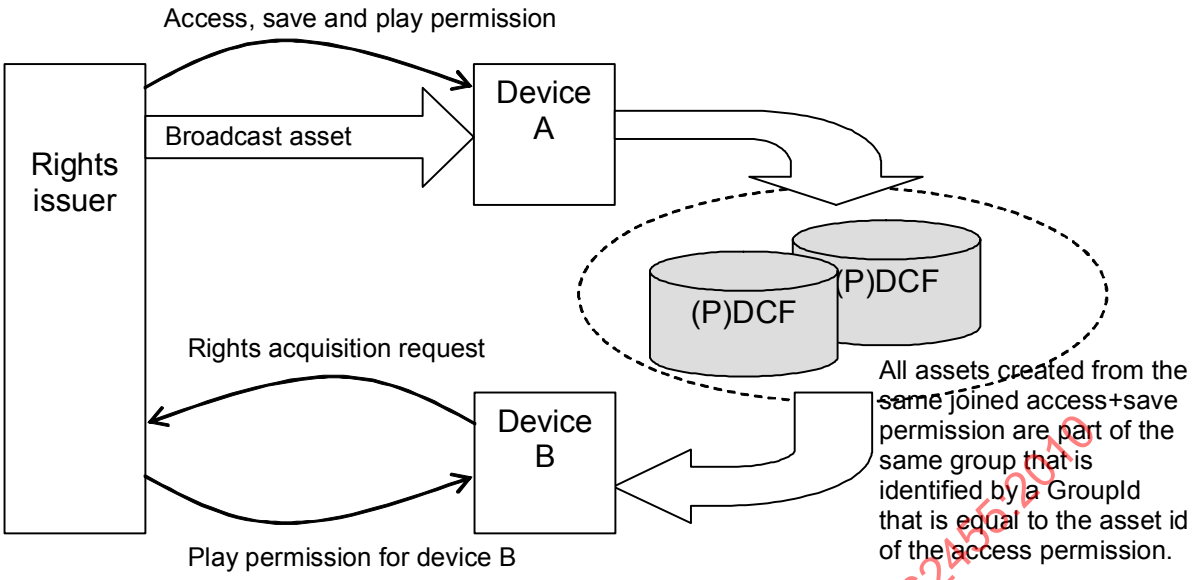


Figure A.18 – Recording and super-distributing the recorded asset

Figure A.18 illustrates the process of recording and super-distribution of assets and the subsequent acquisition of rights for that content.

A.24.3 Specification of the save permission

A.24.3.1 General

The normative statements in this clause only apply to the concept of creating super-distributable OMA assets containing a recording of broadcast content that is suitable for standard OMA DRM 2.0 devices.

A rights issuer can allow a device to make super-distributable recordings of a broadcast asset by including a save permission in a rights object for that asset. The save permission explicitly allows creating new assets containing a rendering of the broadcast content in permanent storage. The device shall also have access permission for that broadcast asset in order to create this permanent copy.

The super-distributable recorded assets shall be in a DCF or PDCF format and are super-distributable to other devices. The recording device shall create a new CommonHeaders box for use in each new asset file. The ContentID and RightsIssuerURL are generated from information that is retrieved from the service guide, and the secure DRM time of the device.

If the device does not support secure DRM time, it shall not grant save permissions.

The context of the broadcast asset (service guide, session description protocol or key stream messages) should provide at least the content identifier, RightsIssuerURL and content encryption key to use when creating the CommonHeaders box and the protected content in each created asset file.

A.24.3.2 Element <save>

Semantics of the save element is defined as shown in Table A.21.