
IPDC Services Purchase and Protection Specification

THIS IS A PROVISIONAL DVB DOCUMENT. IT MAY BE CHANGED BEFORE FINAL ADOPTION BY DVB. THIS PROVISIONAL DOCUMENT IS FOR DISCUSSION PURPOSES ONLY. IMPLEMENTERS ARE NOT ENTITLED TO RELY ON THIS PROVISIONAL DOCUMENT. WHERE POSSIBLE, ITEMS FOR WHICH CONSENSUS HAS NOT BEEN REACHED ARE SUITABLY MARKED, FOR EXAMPLE BY SQUARE BRACKETS. IMPLEMENTERS SHOULD ALSO NOTE THAT ONLY FINAL SPECIFICATIONS ADOPTED BY DVB ARE (SUBJECT TO THE "NEGATIVE DISCLOSURE" RIGHTS OF MEMBERS) ENTITLED TO THE IPR LICENSING TERMS OF DVB'S MEMORANDUM OF UNDERSTANDING.

Contents

1	Scope	9
2	References	10
3	Definitions, symbols and abbreviations	12
3.1	Definitions	12
3.2	Symbols	13
3.3	Abbreviations	13
4	System Overview (informative)	16
4.1	General description of the system and elements	16
4.1.1	Selected Technologies	16
4.1.2	Overview of Operation	18
4.2	The End-to-End System	19
4.3	Modes of Operation and Types of Device	20
4.3.1	Unconnected Devices	20
4.3.2	Scalability Considerations	21
4.4	Purchase Steps	22
4.5	Consumption Steps	25
4.6	Service Protection vs. Content Protection	27
5	Theory of Operation	29
5.1	End-to-end Architecture	29
5.1.2	Special Cases	30
5.1.2.1	Free-To-Air Services	30
5.1.2.2	Free-To-View Services	30
5.2	Electronic Service Guide and Purchase	31
5.3	Registration	32
5.3.1	Concept of the RI context	32
5.3.2	Registration for interactive mode of operation	32
5.3.3	Registration for broadcast (only) mode of operation	32
5.3.4	Mixed-mode registration for interactive and broadcast modes of operation	34
5.4	The Four Layer Model	35
5.4.1	Key Hierarchy	35
5.4.1.1	Keys on the Traffic Layer	35
5.4.1.2	Keys on the Key Stream Layer	35
5.4.1.2.1	Service based subscription	35
5.4.1.2.2	Pay-per view based and service based subscription	36
5.4.1.2.3	Pay-per view based consumption	37
5.4.1.3	Keys on the Rights Management Layer (Broadcast mode)	37
5.4.1.4	Keys on the Rights Management Layer (Interactive mode)	38
5.4.1.5	Keys on the Registration Layer (Broadcast mode)	38
5.4.1.6	Authentication overview	39
5.4.1.6.1	Authentication keys on traffic layer	40
5.4.1.6.2	Authentication keys on key stream layer	40
5.4.1.6.3	Authentication keys on rights management layer (broadcast mode)	41
5.4.1.6.4	Authentication keys on registration layer (broadcast mode)	41
5.5	Deployment for interactive mode of operation	41
5.5.1	Concept of Domains – OMA DRM 2.0 Domains	41
5.6	Deployment for broadcast mode of operation	41
5.6.1	Concept of Domains – local domains	41
5.6.2	Addressing (group / subset / device / domain)	42
5.6.2.1	Addressing the unique group	42
5.6.2.2	Addressing a broadcast group	43
5.6.2.3	Addressing a unique device	43
5.6.2.4	Addressing a local domain	44
5.6.3	Zero Message Broadcast Encryption scheme	44
5.7	Interoperability with Alternative Implementations of the Functionality of Rights Management Layer and Registration Layer	46

6	The Four-Layer Model for Service and Content Protection.....	48
6.1	Traffic Layer.....	48
6.1.1	IPsec.....	48
6.1.1.1	Selectors.....	49
6.1.1.2	Encapsulation Protocol and mode.....	49
6.1.1.3	Encryption Algorithm.....	49
6.1.1.4	Authentication Algorithm.....	50
6.1.1.5	Security Association Management.....	50
6.1.2	SRTP.....	50
6.1.2.1	Key Management.....	51
6.1.2.2	Encryption Algorithm.....	52
6.1.2.3	Authentication Algorithm.....	52
6.2	Key Stream Layer.....	52
6.2.1	Key Stream Message (KSM).....	52
6.2.1.1	Descriptors for access_criteria_descriptor_loop.....	53
6.2.1.2	Constants.....	54
6.2.1.3	Coding and Semantics of Attributes.....	54
6.2.2	Key Stream Discovery.....	59
6.3	Rights Management Layer.....	60
6.3.1	Requirements for Service ROs.....	60
6.3.2	Requirements for Programme ROs.....	61
6.3.3	Delivery of ICROs over Interactivity Channel.....	61
6.3.4	Delivery of BCROs over Broadcast Channel.....	62
6.3.4.1	Broadcast of BCRO Objects.....	62
6.3.4.2	Format of a Broadcast Rights Object (BCRO).....	62
6.3.4.2.1	Format of the asset object.....	64
6.3.4.2.2	Format of the permission object.....	66
6.3.4.2.3	Format of the action object.....	67
6.3.4.2.3.1	Action definitions.....	67
6.3.4.2.4	Format of the constraint object.....	68
6.3.4.2.4.1	Count constraint descriptor.....	68
6.3.4.2.4.2	Timed count constraint descriptor.....	69
6.3.4.2.4.3	Date-time constraint descriptor.....	69
6.3.4.2.4.4	Interval constraint descriptor.....	70
6.3.4.2.4.5	Accumulated constraint descriptor.....	70
6.3.4.2.4.6	Individual constraint descriptor.....	70
6.3.4.2.4.7	System constraint descriptor.....	71
6.3.4.2.4.8	Metering constraint descriptor.....	71
6.4	Registration Layer.....	72
6.4.1	RI Context.....	72
6.4.2	Interactive mode of operation.....	72
6.4.3	Broadcast mode of operation.....	72
6.4.3.1	Protocol overview.....	72
6.4.3.2	offline Notification of Detailed Devicedata protocol.....	73
6.4.3.3	offline Notification of Short Devicedata protocol.....	74
6.4.3.3.1	Request re-registration (only at same RI).....	76
6.4.3.3.2	Request join domain.....	76
6.4.3.3.3	Request leave domain.....	76
6.4.3.3.4	Token consumption report.....	76
6.4.3.3.4.1	Token consumption data definition.....	77
6.4.3.3.5	Notify DRM time drift.....	77
6.4.3.4	1-pass binary Push Device Registration Protocol.....	77
6.4.3.5	1-pass binary Inform Registered Device Protocol.....	78
6.4.3.5.1	force re-registration.....	78
6.4.3.5.2	update RI certificate.....	79
6.4.3.5.3	update DRM_Time.....	79
6.4.3.5.4	update contact number.....	79
6.4.3.5.5	force to join a domain.....	79
6.4.3.5.6	force to leave a domain.....	80
6.4.3.5.7	update a domain.....	80
6.4.3.6	Unique Device Number (UDN) protocol.....	80
6.4.3.6.1	Message syntax.....	81

6.4.3.7	Binary messages	81
6.4.3.7.1	Device data – device_data_inform() message.....	81
6.4.3.7.1.1	Message description	81
6.4.3.7.1.2	Message syntax	82
6.4.3.7.2	Registration data – device_registration_response() message.....	82
6.4.3.7.2.1	Message description	82
6.4.3.7.2.2	Protection of the keyset	88
6.4.3.7.2.3	Message syntax	90
6.4.3.7.3	(Force to) Re-register - re_register_msg() message.....	91
6.4.3.7.3.1	Message description	91
6.4.3.7.3.2	Message syntax	94
6.4.3.7.4	Update RI certificate - update_ri_certificate_msg() message	94
6.4.3.7.5	Updating the DRM time - update_drmtime_msg() message.....	94
6.4.3.7.5.1	Message description	94
6.4.3.7.5.2	Message syntax	96
6.4.3.7.6	Update the contact number – update_contact_number_msg() message.....	96
6.4.3.7.6.1	Message description	96
6.4.3.7.6.2	Message syntax	99
6.4.3.7.6.2.1	Format of the contact object	99
6.4.4	Domain joining and leaving	100
6.4.4.1	protocol overview	101
6.4.4.2	offline Domain Join Request	101
6.4.4.3	offline Domain Leave Request	101
6.4.4.4	Binary messages	102
6.4.4.4.1	Domain data - domain_registration_response() message.....	102
6.4.4.4.1.1	Message description	102
6.4.4.4.1.2	Protection of the keyset	105
6.4.4.4.1.3	Message syntax	107
6.4.4.4.2	Updating a domain - domain_update_response() message	108
6.4.4.4.2.1	Message description	108
6.4.4.4.2.2	Message syntax	111
6.4.4.4.3	(Force to) Join a domain - join_domain_msg() message	112
6.4.4.4.6	(Force to) Leave a domain - leave_domain_msg() message.....	112
6.4.4.4.6.1	message syntax	112
6.4.5	Token handling.....	113
6.4.5.1	Protocol overview.....	113
6.4.5.2	token request protocol	113
6.4.5.3	token reporting protocol	113
6.4.5.4	Binary messages	113
6.4.5.4.1	delivering tokens – token_delivery_response() message	113
6.4.5.4.1.1	Message description	113
6.4.5.4.1.3	Message syntax	118
7	Rights Issuer Services	120
7.1	Expected Mode of Operation (Informative).....	120
7.2	Scheduled RI Stream	121
7.3	Ad-hoc RI Stream	121
7.4	In-Band RI Streams within a Media Service	122
7.5	Broadcast Format of RI Streams.....	122
7.5.1	IP Characteristics	122
7.5.2	RI Stream Packet Format	122
7.5.3	Implementation Notes	124
7.5.3.1	Unreliable Delivery	124
7.5.3.2	Changes in Packet Order (Informative).....	124
7.5.3.3	Addressing of Objects	124
7.6	Mapping of Messages to RI Services and Streams	124
7.6.1	Rights Issuer Services With Complete Schedule Information	125
7.6.2	Rights Issuer Services Without Complete Schedule Information	125
7.7	Discovery of RI Services, Streams and Schedule Information	125
7.8	Certificate Chain Updates	125
7.9	Resending of BCROs.....	126
7.9.1	Resending of BCROs to Interactive Devices	126

7.9.2	Resending of BCROs to Broadcast Devices	126
7.10	Summary of Requirements for Rights Issuers	126
7.11	Summary of Requirements for Devices	127
8	Service Subscription and Purchase	128
8.1	Purchase over the Interactivity Channel	129
8.1.1	Typical Purchase Sequences	129
8.1.1.1	Bulk Download of Service and Program Keys	129
8.1.1.2	Bulk Download of Purchase Information	131
8.1.1.3	Announcement of Purchase Items in Service Guide	133
8.1.1.4	Pricing Inquiry	134
8.1.1.5	Unsuccessful Purchase	136
8.1.1.6	Successful Purchase	140
8.1.1.7	Subscription RO Renewal and Asynchronous Charging	145
8.1.1.8	Asynchronous Charging and Cancellation of Open-Ended Subscriptions	149
8.1.1.9	Purchase of Tokens for Consumption-based Charging	150
8.1.2	Protocol	153
8.1.2.1	HTTP Headers	153
8.1.2.2	Signatures	153
8.1.3	XML Schemas for Request and Response Messages	153
8.1.3.1	Basic Types	153
8.1.3.1.1	User Data Type	153
8.1.3.1.2	Device Data Type	153
8.1.3.1.3	Domain Type	154
8.1.3.1.4	ServiceOperationCentreType	154
8.1.3.1.5	PriceType	155
8.1.3.1.6	Purchase Item Type	155
8.1.3.1.7	Request Type	155
8.1.3.1.8	Response Type	156
8.1.3.2	Error Codes	157
8.1.3.3	Pricing Request	159
8.1.3.3.1	XML Schema	159
8.1.3.3.2	Example	159
8.1.3.4	Pricing Response	160
8.1.3.4.1	XML Schema	160
8.1.3.4.2	Example: Successful Pricing Response	160
8.1.3.5	Purchase Request	161
8.1.3.5.1	Schema	161
8.1.3.5.2	Example	161
8.1.3.6	Purchase Response	162
8.1.3.6.1	XML Schema	162
8.1.3.6.2	Example: Successful Purchase Response with RO Acquisition Trigger	162
8.1.3.6.3	Example: Unsuccessful Purchase Response with Registration Trigger	162
8.1.3.6.4	Example: Unsuccessful Purchase Response with Purchase-Item-specific Error	163
8.1.3.7	Subscription RO Renewal Request	163
8.1.3.7.1	XML Schema	163
8.1.3.7.2	Example	163
8.1.3.8	Subscription RO Renewal Response	164
8.1.3.8.1	Schema	164
8.1.3.8.2	Example: Successful Renewal Response with RO Acquisition Trigger	164
8.1.3.8.3	Example: Unsuccessful Renewal Response with Registration Trigger	164
8.1.3.8.4	Example: Unsuccessful Renewal Response with Purchase-Item-specific Error	165
8.1.3.9	Subscription Cancellation Request	165
8.1.3.9.2	Example	165
8.1.3.10	Subscription Cancellation Response	166
8.1.3.10.1	Schema	166
8.1.3.10.2	Example: Successful Cancellation Response	166
8.1.3.10.3	Example: Unsuccessful Cancellation Response With Purchase-Item-specific Error	166
8.1.3.11	Token Request	167
8.1.3.11.1	XML Schema	167
8.1.3.11.2	Example	167
8.1.3.12	Token Response	167

8.1.3.12.1	Schema.....	167
8.1.3.12.2	Example: Successful Token Response.....	168
8.1.3.12.3	Example: Unsuccessful Token Response.....	168
8.2	Purchase for Mixed-Mode Devices	168
8.3	Out-of-Band Purchase.....	168
8.3.1	Means of purchase (informative)	168
8.3.2	Out-of-Band purchase from service guide data (normative)	169
8.4	Required Service Guide Information	170
8.4.1	Service Operation Centre	172
8.4.2	Customer Operation Centre.....	172
8.4.3	Service.....	173
8.4.4	ScheduleItem.....	173
8.4.5	ContentItem.....	173
8.4.6	Purchase Item.....	174
8.4.7	Purchase Data.....	174
9	Head-end for Service Protection (INFORMATIVE)	175
9.1	Function Blocks	175
9.2	Considerations	175
9.3	Proposed Network Elements.....	177
10	Deployment and Roaming Scenarios (INFORMATIVE)	178
10.1	Deployment Option A (RI as part of SOC).....	179
10.1.1	Local Scenario.....	179
10.1.2	Roaming Scenario	180
10.1.3	Conclusion to Deployment Option A.....	181
10.2	Deployment Option B (RI as part of COC)	182
10.2.1	Local Scenario.....	182
10.2.2	Roaming Scenario	183
10.2.3	Conclusion to Deployment Option B	184
10.3	Conclusion to Roaming Scenarios	184
Annex A	(normative):.....	185
A.1	Security Considerations	185
A.1.1	Handling weak keys	185
A.1.2	Handling OCSP grace period	185
A.2	Status and Error Message Handling.....	186
A.3	Mapping of messages to DVB-H Time Sliced Bursts	188
A.3.1	Key Stream Messages	188
A.3.2	RI Service Channel	188
A.3.2.1	In-Band RI Stream	188
A.4	Conversion between time and date conventions	188
A.4.1	local time offset.....	190
A.5	Conversion of OMA DRM 2.0 content identifiers to binary content identifiers.....	190
A.6	Conversion of OMA X509PKISHash values to binary values	191
A.7	Limits of the surplus_block().....	191
A.7.1	Standard keyset	191
A.7.2	Extended keyset	192
A.8	Function F _{ZMB}	193
A.8.1	Definitions.....	193
A.8.2	Examples.....	195
A.8.2.1	Sample tree and keyset	195
A.9	Authentication.....	195
A.9.1	Authentication for IPsec.....	195
A.9.2	Authentication for KSMS	196
A.9.2.1	Transport of SEAK and PEAK in OMA DRM 2.0 Rights Objects.....	196
A.9.2.2	Transport of SEAK and PEAK in BCROs	197
A.9.3	Authentication of BCROs	197
A.9.4	General authentication mechanism	197
A.9.5	Authentication of token delivery response messages	198
A.10	Checksum algorithms	198
A.10.1	UDN checksum	198
A.10.2	ARC checksum	200

A.11	RSA signatures under PKCS#1	201
A.12	C-style types	201
A.13	Tag Length Format for keyset_block.....	202
A.13.1	TLF syntax definition.....	202
A.13.2	TLF examples	203
A.13.3	LLDF syntax	204
A.14	Message_tag overview.....	204
A.15	Authentication of the tokens_consumed field in the token consumption data.....	204
A.16	Management of tokens by RIs and devices.....	205
A.16.1	Token management by RIs.....	205
A.16.1.1	Pre-paid token business model	205
A.16.1.2	Post-paid token business model.....	206
A.16.1.3	Switching from the pre-paid token business model to the post-paid token business model	206
A.16.1.4	Switching from the post-paid token business model to the pre-paid token business model	206
A.16.1.5	Stopping the post-paid token business model.....	207
A.16.2	Token management by devices	207
A.17	Conformance Table.....	208
Annex B (informative):	211
B.1	Service Purchase Scenarios.....	211
B.1.1	Free-to-Air (unscrambled).....	211
B.1.2	Free-to-View (scrambled)	211
B.1.3	Subscription-based services	211
B.1.3.1	Using the Interactivity Channel.....	211
B.1.3.2	Using the Broadcast Channel	211
B.1.4	One-Off Purchases	211
B.1.5	Bundling with a mobile subscription.....	212
B.1.6	Online and Offline control of content	212
B.1.7	Billing and Charging types.....	212
B.1.7.1	On line	212
B.1.7.2	Off line	212
B.1.7.3	Pre-Paid	212
B.1.7.4	Post-Paid.....	213
B.1.7.5	Tokens	213
B.1.8	Bundles of Services.....	213
B.1.9	Free Preview	214
B.1.9.1	Free-to-Air Preview	214
B.1.9.2	Free Preview with Rights Objects	214
B.1.9.3	Free Preview with Access Criteria	214
B.1.9.4	Timed Preview.....	214
B.1.9.5	Preview with Cryptographically-Enforced Period.....	214
B.1.9.6	Advertising Previews via the ESG	214
B.2	Operative Scenarios	215
B.2.1	Revocation	215
B.2.2	Expiration.....	215
B.2.2.1	Expiration of RI Context	215
B.2.2.2	Expiration of a BCRO	215
B.2.3	Moving from one group to another	216
B.3	Scalability, Bandwidth Considerations.....	216
B.4	Use of the Interoperability Point.....	218
B.5	Security Considerations	220
B.5.1	Threat Mapping	222
History	227

1 Scope

The present document specifies a scheme for IPDC Services Purchase and Protection, in order that the user of a broadcast and/or interactive device can consume services with high value content, protected by the horizontal OMA DRM 2.0 system or with another DRM system. This document specifies the protocols and messages that are needed to decrypt protected services, to protect and use keys over different layers of the service protection model as well as the purchase and consumption of the services.

The content of the document is normative, except for those sections that are labelled “informative”.

It is expected that the IPDC Services Purchase and Protection will be used in mobile consumer equipment.

2 References

- [AES_WRAP] NIST Key Wrap, National Institute of Standards and Technology, 16 November 2001
- [EN 300 468] ETSI EN 300 468, Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems
- [EN 301 192] ETSI EN 301 192, Digital Video Broadcasting (DVB); DVB specification for data broadcasting
- [EUROCRYPT] EN 50094:1992 - CLC/TC 206 Access control system for the MAC/packet family: EUROCRYPT, 1992
- [FIAT_NAOR] Broadcast encryption, Amos Fiat and Moni Naor, 1998
- [FIPS 180.2] National Institute of Standards and Technology. Secure Hash Standard (SHS). FIPS 180-2. August 1, 2002
- [FIPS 197] National Institute of Standards and Technology, Specification for the Advanced Encryption Standard (AES) FIPS 197. November 26, 2001
- [FIPS 198] The Keyed-Hash Message Authentication Code (HMAC), Information Technology Laboratory National Institute of Standards and Technology, Gaithersburg, MD 20899-8900, Issued March 6, 2002
- [ISO639] ISO 639, Codes for the representation of names of languages. ISO, 1988
- [ISO4217] ISO 4217, Standards for Currency Names. ISO, 1999
- [NIST 800-38A] NIST Special Publication 800-38A, 2001 Edition, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, Issued December 2001
- [OBEX] IrDA Object Exchange Protocol (OBEX), Version 1.3, January 2003
- [OMA-DRM-ARCH] DRM Architecture v2.0, Open Mobile Alliance
- [OMA-DRM-DRM] DRM Specification v2.0, Open Mobile Alliance
- [OMA-DRM-REL] DRM Rights Expression Language v2.0, Open Mobile Alliance
- [PKCS#1] PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002
- [RFC 768] RFC 768, User Datagram Protocol, J. Postel – ISI. 28th August 1980
- [RFC 1738] RFC 1738, Uniform Resource Locators (URL), T. Berners-Lee - CERN, L. Masinter - Xerox Corporation, M. McCahill - University of Minnesota, December 1994
- [RFC 2104] RFC 2104, HMAC: Keyed-Hashing for Message Authentication. H. Krawczyk, M. Bellare, R. Canetti. February 1997
- [RFC 2401] RFC 2401, Security Architecture for the Internet Protocol. S. Kent, R. Atkinson. November 1998
- [RFC 2404] RFC 2404, The Use of HMAC-SHA-1-96 within ESP and AH. C. Madson, R. Glenn. November 1998
- [RFC 2406] RFC 2406, IP Encapsulating Security Payload (ESP). S. Kent, R. Atkinson. November 1998
- [RFC 2451] The ESP CBC-Mode Cipher Algorithms. R. Pereira, R. Adams. November 1998
- [RFC 3566] RFC 3566, The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec, S. Frankel (NIST) H. Herbert (Intel), September 2003

- [RFC 3602] RFC 3602, The AES-CBC Cipher Algorithm and Its Use with IPsec. S. Frankel, R. Glenn, S. Kelly. September 2003
- [RFC 3629] RFC 3629, UTF-8, a transformation format of ISO 10646. F. Yergeau, November 2003.
- [RFC 3664] RFC 3664, The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE), P. Hoffman VPN Consortium, January 2004
- [RFC 3711] RFC 3711, The Secure Real-time Transport Protocol (SRTP). M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman. March 2004
- [SCHNEIER] Applied Cryptography, Bruce Schneier, Second Edition, John Wiley & Sons, Inc, 1996
- [VERHOEF_1969] Verhoef J, Error detecting decimal codes, Mathematical Centre Tract 29, The mathematical Centre, Amsterdam, 1969
- [XML_XCAN] Exclusive XML Canonicalization: Version 1.0, John Boyer, Donald E. Eastlake 3rd and Joseph Reagle, W3C Recommendation 18 July 2002

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the following definitions apply:

Broadcast Channel: a broadcast network (see below).

Local domain: a group of devices for which Rights Objects granting the same rights to the whole group can be issued via the broadcast channel.

Broadcast Network: a digital transmission supporting only unidirectional communication from the broadcaster to the device.

Broadcast Rights Object: a content, programme or service Rights Object delivered over the broadcast network.

Broadcast Service: a service carried over a broadcast network.

Broadcast Device: a device which is capable of receiving protected broadcasts but which cannot access an interactivity channel. All messages sent to such a device are delivered via the broadcast channel, and all communication from the device to Rights Issuers is done out of band.

Content: any form of digital media which can be acquired and presented by a device.

Content Protection: the protection of content such that it can only be presented by authorised devices.

Content Rights Object: a Rights Object containing a content key and pertaining to content protection.

Device: an item of equipment capable of receiving services.

DRM Time: a secure, non user-changeable time source. The DRM time is measured in the UTC time scale.

Electronic Service Guide: used in IP Datacast systems to signal the services that are available; how they can be received and what they contain.

Interactive Device: a device that is capable of directly connecting to a Rights Issuer using an appropriate protocol over an appropriate transport/network layer interface, e.g. HTTP over TCP/IP.

Interactive Domain: an OMA DRM 2.0 domain, as specified in [OMA-DRM-DRM]. This is a group of devices for which Rights Objects can be issued via the interactivity channel granting rights to the whole group.

Interactivity Channel Rights Object: a content, programme or service Rights Object delivered over the interactivity channel.

Interactivity Channel: an interactivity network (see below).

Interactivity Network: a network supporting bi-directional communication and the reliable delivery of messages between the device and the Rights Issuer.

Interactivity Network Cell: a cell forming part of an interactivity network.

IP Flow: a stream of IP datagrams each sharing the same IP source and destination address.

Key Stream Message: a message broadcast alongside a protected service, carrying key material to decrypt and optionally authenticate the service.

Metering Rights Object: a content, programme or service Rights Object used in conjunction with consumption-based charging.

Nonce: A randomly chosen value, different from previous choices, inserted in a message to protect against replay attacks.

Programme: a logical portion of a service with a distinct start and end.

Programme Rights Object: a Rights Object containing a programme key or keys and pertaining to service protection.

Rights Issuer: an entity that registers devices and provides Rights Objects allowing devices to receive protected services.

Rights Issuer Context: a Rights Issuer Context consists 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. This Rights Issuer Context is necessary for a device to successfully participate in all the protocols of the ROAP suite, except the Registration Protocol.

Rights Issuer Service: an IP Datacast channel used to carry Broadcast Rights Objects, registration data and other messages from a Rights Issuer over a broadcast channel.

Rights Object: a collection of permissions, keys and other attributes which are linked to items of content or services.

Service: one or more IP flows intended to be presented together. Examples include, but are not limited to, audio and video services.

Service Guide: the Electronic Service Guide (see above).

Service Operation Centre: the service provider (see below).

Service Protection: protection of a service such that only authorised devices are able to receive and decode it.

Service Provider: the provider of a broadcast service – the entity that broadcasts the service, and provides key and schedule information to Rights Issuers.

Service Rights Object: a Rights Object containing one or multiple service keys and pertaining to service protection

Token Rights Object: a 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

Unconnected Device: a device, as defined in [OMA-DRM-ARCH], that cannot directly connect to a Rights Issuer via an interactivity channel for the acquisition of Rights Objects, but can do so via an intermediary device.

User: The person using the device to receive protected services.

3.2 Symbols

For the purposes of the present 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
$A \oplus B$	Bit-wise exclusive OR of A and B
$A \mid B$	Bit-wise OR of A and B
$A \parallel B$	Concatenation of A and B
$A \ll B$	Shiftwise left operation of A and B
$A \gg B$	Shiftwise right operation of A and B
LSB _m (X)	The bit string consisting of the <i>m</i> least significant bits of the bit string X.
MSB _m (X)	The bit string consisting of the <i>m</i> most significant bits of the bit string X.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AES	Advanced Encryption Standard
ARC	Action Request Code
BAK	BCRO Authentication Key
BCD	Binary Coded Decimal
BCI	Binary Content ID
BCRO	Broadcast Rights Object
BGK	Broadcast Group Key
BOC	Broadcast Operation Centre
CEK	Content Encryption Key
CI	Content Issuer
CID	Content ID
COC	Customer Operation Centre
CPCM	Content Protection and Copy Management
DCF	DRM Content Format
DEK	Deduced Encryption Entitlement Key
DIST Mgmt	Service Distribution Management
DRD	Device Registration Data
DRM	Digital Rights Management
DVB	Digital Video Broadcasting
ESG	Electronic Service Guide
ESP	Encapsulating Security Payload
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HMAC	Hashed Message Authentication Code
i-point	Interoperability Point
ICRO	Interactivity Channel Rights Object
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
IPDC	Internet Protocol Datacasting
IPsec	IP Security
IRD	Inform Registered Device (protocol)
KSM	Key Stream Message
LDK	Local Domain Key
LLDF	Long-form Local Domain Filter (a.k.a. longform_domain_id)
MAC	Message Authentication Code
MJD	Modified Julian Date
MK	Master Keys
MKI	Master Key Index
NOC	Network Operation Centre
NDD	Notification of Detailed Data
NSD	Notification of Short Data
OCSP	Online Certificate Status Protocol
OMA	Open Mobile Alliance
OTA	Over The Air (i.e. transfer over a wireless connection)
PAK	Programme Authentication Key
PAS	Programme Authentication Seed
PDCF	Packetized DRM Content Format
PDR	Push Device Registration
PEAK	Programme Encryption / Authentication Key
PEK	Programme Encryption Key
PKC	Public Key Certificate
PKC-ID	PKC Identifier: the hash of the Public Key Certificate
PKI	Public Key Infrastructure
REL	Rights Expression Language
RI	Rights Issuer
RIAK	Right Issuer Authentication Key
RO	Rights Object
ROAP	Rights Object Acquisition Protocol
ROC	Roll-Over Counter
ROT	Root Of Trust
RSA	Rivest-Shamir-Adelman public key algorithm
RTP	Real Time Protocol

SA	Security Association
SAK	Service Authentication Key
SAS	Service Authentication Seed
SEAK	Service Encryption / Authentication Key
SEK	Service Encryption Key
SHA-1	Secure Hash Algorithm
SLDF	Short-form Local Domain Filter (a.k.a. shortform_domain_id)
SMS	Short Message Service
SOC	Service Operation Centre
SPI	Security Parameters Index
S RTP	Secure Real-time Transport Protocol
SUB Mgmt	Service Subscription Management
TAK	Traffic Authentication Key
TAS	Traffic Authentication Seed
TDK	Token Delivery Key
TEK	Traffic Encryption Key
UDF	Unique Device Filter
UDK	Unique Device Key
UDN	Unique Device Number
UDP	User Datagram Protocol
UGK	Unique Group Key
URI	Uniform Resource Identifier
WAP	Wireless Application Protocol
WIM	WAP Identify Module
ZMB	Zero Message Broadcast

4 System Overview (informative)

4.1 General description of the system and elements

This specification describes a service protection system for services transmitted over an IP Datacast (IPDC) infrastructure. It enables access to services to be restricted to authorised users.

The solution is specific to IP Datacast channels. It can operate on either:

- The Internet Protocol (IP) level, based on the IPsec security standard, in which case it is transparent to IP based applications (such as video players); or
- The transport layer, using SRTP. 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 IP 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 solution extends that to the case of receiving streaming content over the broadcast channel. It also provides a means of performing rights management over an broadcast channel.

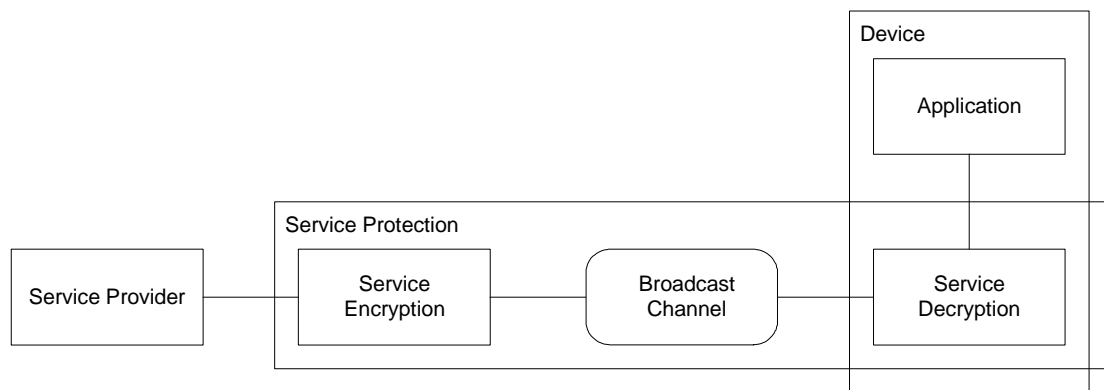


Figure 1: System Overview

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 provides an alternative for protecting content at the transport layer.

At every level of this specification, special consideration has been given to reducing the power requirements of receiving devices.

4.1.1 Selected Technologies

These are the main standards on which the solution is based:

- Advanced Encryption Standard (AES, see [FIPS 197]) in the Cipher Block Chaining mode with 128 bit keys, for actual content encryption. Furthermore, OMA DRM uses AES-WRAP in its Rights Objects and optionally AES CBC-MAC.
- Secure Internet Protocol (IPsec, see [RFC2406]) using the Encapsulating Security Payload (ESP) protocol, for implementing service encryption and decryption as a function of the IP stack. Only transport mode is used.
- Secure Real Time Protocol (SRTP, see [RFC3711]) for alternatively implementing service protection at the transport layer. SRTP uses AES-CM (counter mode).
- A traffic key delivery protocol and management as specified in this document.

- Open Mobile Alliance (OMA) Digital Rights Management version 2.0 (OMA DRM 2.0, see [OMA-DRM-DRM]) for managing rights to services, the associated service keys and the cryptographic protection of those keys themselves. This specification introduces some adaptations to OMA DRM 2.0 for IPDC Service Protection.
- Rights object delivery and device registration over a broadcast channel, without make any use of an interactivity channel, are also specified in this document.
- Adapted Zero Message Broadcast according to [FIAT NAOR] in 1-resilient mode.

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 which 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 that 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.
- The traffic key (i.e. the actual content encryption key) management framework and protocol are specified in this document. The efficiency and robustness of the solution is achieved by a particular delivery protocol and management scheme for the frequently changing traffic keys.
- Among the various rights management alternatives, OMA DRM 2.0 is the one which makes IPDC a part of the same value chains which will be used for selling content and services in the cellular world, and which thus will be implemented by many devices in any case. OMA DRM, too, has existing applications.
- 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 optimise the use of the broadcast channel, some new standardisation 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, thereby greatly saving on bandwidth. 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) we will need:

- device transmission and storage for $2\log(m)$ Broadcast Group Keys (a.k.a. “BGK”s).
- device computing power to derive $(m-1)$ leaf keys.
- device computing power to process, worst case, $(m-1)$ leaf keys to create the “DEK” key that covers the SEK.

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

An alternative like the Naor Noar Lotspiech (NNL) scheme uses more bandwidth by potentially saving slightly on local device processing requirements, but this is opposite to the above-mentioned consideration.

Fiat Noar is used in 1-resilient mode, thereby making a collusion attack in theory possible. Making the scheme k -resilient, as discussed in [FIAT_NAOR] will increase the number of keys dramatically. The potential threat of 1-resilience is countered by measures described in section 5.6.3.

4.1.2 Overview of Operation

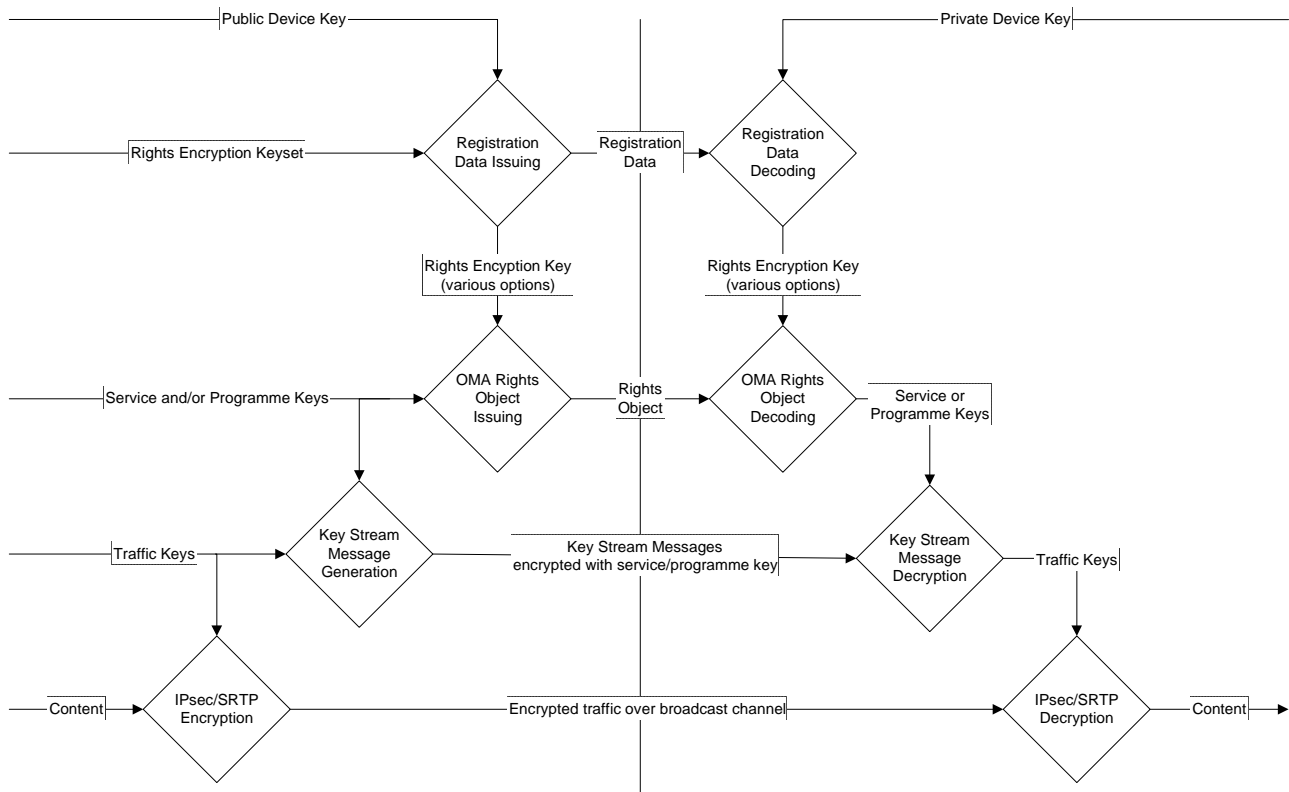


Figure 2: Service protection via four layer model

As illustrated in Figure 2, the solution is based on a four-layer cryptographic architecture, with an optional optimisation 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 as per the SRTP specification. These are used by the IPsec or SRTP 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 which 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 subscribers 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 ESG, 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 and 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 utilised 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 described in this specification.

When delivering Rights Objects over the broadcast channel, bandwidth is a major constraint. This specification addresses this problem in two complimentary 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 described 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 the interactivity or broadcast channels. In the case that the 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 specification defines an efficient and user friendly process for the registration of devices which 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 the 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.

Note that the above assumes the use of OMA DRM 2.0. Should another DRM system be used, the Rights Management Layer and the Registration Layer would change accordingly.

4.2 The End-to-End System

This section briefly describes the major roles within the system. For a more detailed description, see chapter 5.1.

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

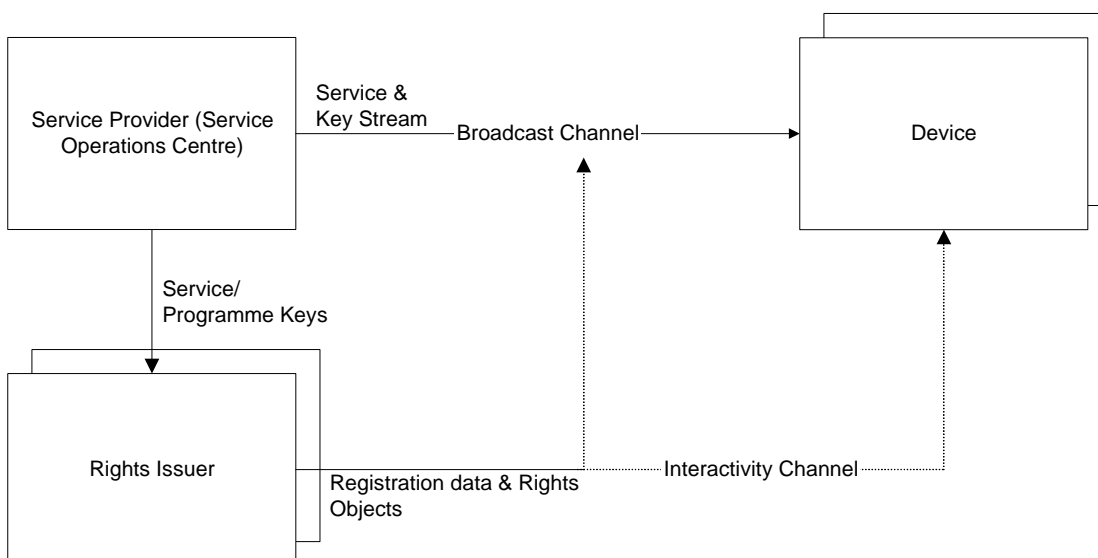


Figure 3: Highly simplified view of the End-to-End system

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 those devices allowing them to decrypt the services which 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.

The interoperability point allows different Rights Issuers to use different DRM systems (or even the same Rights Issuer to use multiple DRM systems) to control access to the same broadcast service, without needing to broadcast the service or the key stream multiple times.

4.3 Modes of Operation and Types of Device

This specification supports populations of “interactive”, “broadcast” and “mixed-mode” devices.

- An *interactive device* supports some form of interactivity channel which can be used to communicate with Rights Issuers, e.g. GPRS. Messages from the device to a Rights Issuer and from the Rights Issuer to a device are sent via the interactivity channel. Services are received via the broadcast channel.
 - Reliable delivery of messages to interactive devices is possible. It is also possible to locate a device to a particular cell of an interactivity network, if this is required, although it should be noted that the location of the interactivity network cell to which a device is connected could be different to the location of the transmitter from which the device receives broadcast data.
- A *broadcast device* does not support an interactivity channel. Requests to Rights Issuers are made by the user via some out-of-band communication, such as a telephone call, an SMS message, a form on a website or some entirely different means. Messages from a Rights Issuer to the device are sent via the broadcast channel.
 - Reliable delivery of messages to broadcast devices may not always be possible, and it is typically not possible to locate a device. Therefore, messages to be delivered to broadcast devices are generally carried across the whole network, and regularly repeated.
- A *mixed-mode device* is also possible. Such a device supports operation both via the interactivity channel and via the broadcast channel.

Should a Rights Issuer use OMA DRM 2.0 to support the Rights Management and Registration Layers, interactive devices can use the existing 4-Pass ROAP from OMA DRM 2.0 [OMA-DRM-DRM] to register with a Rights Issuer and acquire Rights Objects. Some extensions to this scheme are provided in this specification to support the protection of broadcast services.

A new 1-Pass ROAP, called the 1-pass binary Push Device Registration Protocol, is defined for the delivery of registration data and Rights Objects to broadcast devices.

A Rights Issuer can choose to register mixed-mode devices to operate via the interactivity or broadcast channels, or both. It is also possible for the Rights Issuer to signal at any time which method will be used.

4.3.1 Unconnected Devices

[OMA-DRM-DRM] defines so called unconnected devices. These devices do not have direct access to an interactivity channel, but are capable of making a connection via an intermediary interactive device. [OMA-DRM-ARCH] describes a method for these unconnected devices to register with a Rights Issuer as part of a domain, via an intermediary using the [OBEX] protocol. The intermediary can then purchase domain Rights Objects and forward them to the unconnected device, embedded in a DCF. For Rights Objects referring to broadcast services, this DCF can otherwise be empty.

A broadcast device could potentially be able to use the OBEX facility to operate as a mixed-mode device. Furthermore, a device which would not otherwise be able to support this specification could potentially be able to become an interactive device. This leads to the following additional types of device.

- *Unconnected interactive device* (which can receive and decode protected services but lacks the ability to function as a full broadcast device as defined in this specification). As with any interactive device, the unconnected interactive device can be part of a OMA DRM 2.0 interactive domain and can use domain Rights Objects to receive protected services for those interactive domains.
- *Unconnected mixed-mode device* (which is a broadcast device, but can also act as an unconnected device for the acquisition of Rights Objects). The device can form part of two types of domains.

- A local domain which is supported by the BCRO and/or ICRO, in the usual manner.
- An OMA DRM 2.0 interactive domain, which is supported by domain Rights carried in ICROs.

In both cases Rights Objects can be bought over an interactivity channel by an intermediary.

Note: For reasons of completeness: a mixed-mode device is, of course, also capable of buying rights out of band.

Figure 4 shows Rights Issuer communication with various types of device.

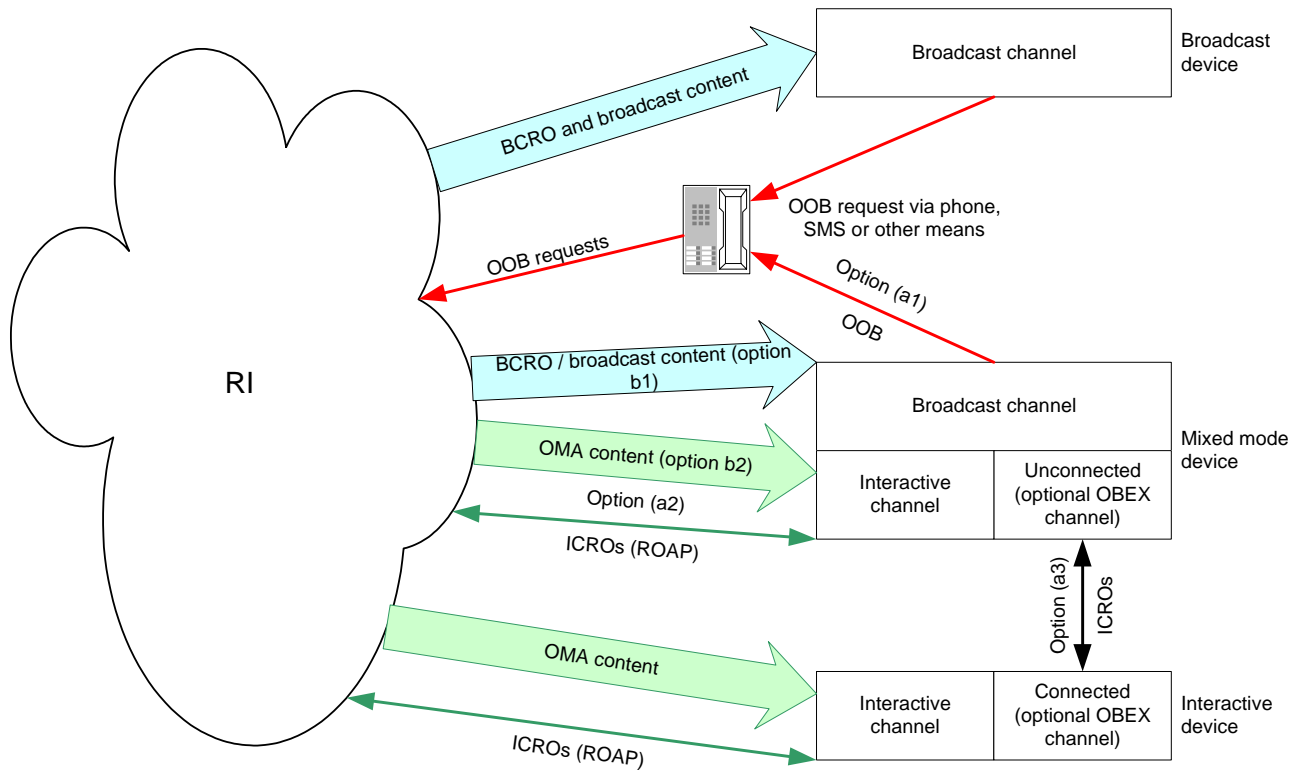


Figure 4: Rights Issuer communication with various types of device

4.3.2 Scalability Considerations

This specification provides considerable scope for Service Providers to balance requirements for efficient bandwidth utilisation with the latency of delivering Rights Objects to a device, in order to build systems that are scalable to many millions of users. It also provides means for minimising the power consumption of devices. The facilities provided for use with OMA DRM 2.0 include:

- The option to provide a service for the spontaneous, ad-hoc delivery of Rights Objects via the broadcast channel.
- The ability to broadcast the same Rights Objects to groups of devices in the same message.
- The ability to broadcast a carousel of Rights Objects, and to provide a schedule for this carousel, so that devices only need to listen to the carousel at the indicated times.
- The ability to broadcast Rights Objects in advance of when they will be used.
- The ability to make use of spare bandwidth that might be available at certain times to deliver Rights Objects.
- The ability for Rights Issuers to decide on appropriate service key life times.
- Rights Objects can be delivered via the broadcast or, when available, the interactivity channel.

Annex B.3 provides a detailed description of bandwidth and scalability considerations.

4.4 Purchase Steps

The customer device purchases a service protected by encryption by obtaining a Rights Object (RO) containing key material from a Rights Issuer. This is carried out with the help of an electronic commerce party, here called E-Commerce System. The fourth player of the purchase is the actual broadcaster of the service, especially its broadcast management which is responsible for generating the key material, which will be used for encrypting the actual service when broadcast.

Please note that for the sake of familiarity, in this informative section vague, undefined terms like “Broadcast Management” and “E-Commerce System” are used for describing functions of the IPDC infrastructure. Later, in the normative sections, they will be replaced by more specific and defined terms such as (respectively) “Service Operation Centre (SOC)” and “Customer Operation Centre (COC)”. The same applies to the term “DRM Agent”, which represents the implementation of the DRM system within the device.

The four parties involved in the purchase steps are shown in Figure 5 below, illustrating the purchase steps in case of an interactive device, capable of communicating with the E-Commerce System and the Rights Issuer across an interactivity channel.

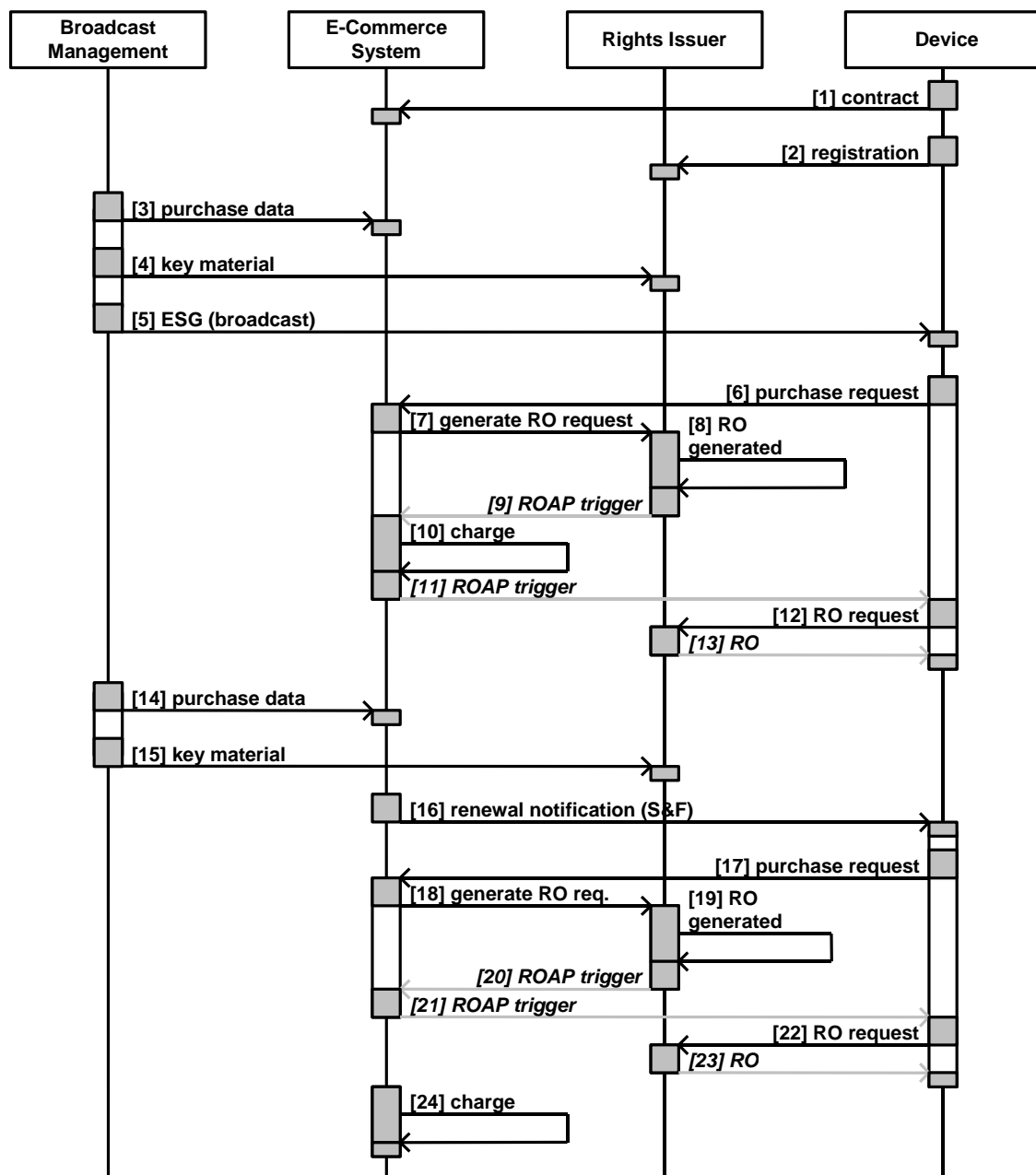


Figure 5: Purchase steps in case of an interactive device

To enable purchase, device needs [1] a contractual relationship with the E-Commerce System, and [2] a DRM registration with the Rights Issuer. Thus both the E-Commerce system and Rights Issuer can authenticate the device whenever it interacts with them. Furthermore, the Rights Issuer knows the public key of the device, and can send key material to it via the interactivity channel in an RO which is encrypted by the public key – only the secure DRM agent in the device can decrypt the key material, using its secret private key.

Now, as the broadcaster prepares to broadcast a service, it distributes information about the service to the other parties: [3] data for purchasing the service to the E-Commerce System, [4] key material for decrypting the service (soon to be broadcast) to the Rights Issuer, and an [5] electronic service guide (ESG) to the device; the ESG is broadcast over the broadcast channel. Proper service identifiers in the distributed material make it possible to identify each service and service bundle, their purchase options, etc.

Based on the ESG, the human user of the device decides to purchase the service.

The device then [6] requests purchase of the service from the E-Commerce System, which [7] requests the Rights Issuer to [8] generate the RO and [9] return ROAP (Rights Object Acquisition Protocol) trigger data for retrieving the RO. The E-Commerce System then [10] charges the device and [11] returns the ROAP trigger to it. The device then uses the ROAP trigger to [12] request the RO from the Rights Issuer, [13] receiving the RO, encrypted by the public key of the device. All these interactions between the device and the network elements use the reliable interactivity channel, and (from the device) are carried out by a protocol like HTTP and HTTPS: the device issues requests and receives responses.

Please note that this specification does not specify the timing of customer charging by the E-Commerce System.

If the service is of a subscription type, eventually both [14] purchase data and [15] key material will change, and the device thus has to obtain the new key material in a renewed RO.

At this point, the E-Commerce System could notify the device of the renewal need by [16] sending a renewal notification to it, typically using some store-and-forward (S&F) mechanism of the interactivity channel, for instance the short message service in GSM networks. Use of the renewal notification is optional however, subject to the [1] contract with the E-Commerce System. Alternatively, the device could detect the need for RO renewal itself.

To renew the RO, the device repeats steps [6] through [13] of the original purchase, illustrated as steps [17] through [23] in Figure 5. Repeated charging of a continuous subscription as step [24] depends on the purchase in question: some continuous services could be charged repeatedly, while time limited services could be charged only once, despite the fact that key material is periodically changed in order to increase the security of service protection.

From the device point of view, due to the purchase steps, its secure DRM agent at all times holds an RO with the key material for decrypting the service, encrypted with the public key of the device, which only the DRM agent can decrypt using the secret private key it holds.

Let us now look at the differences between the above procedure, and the procedure for broadcast devices, illustrated in Figure 6 below.

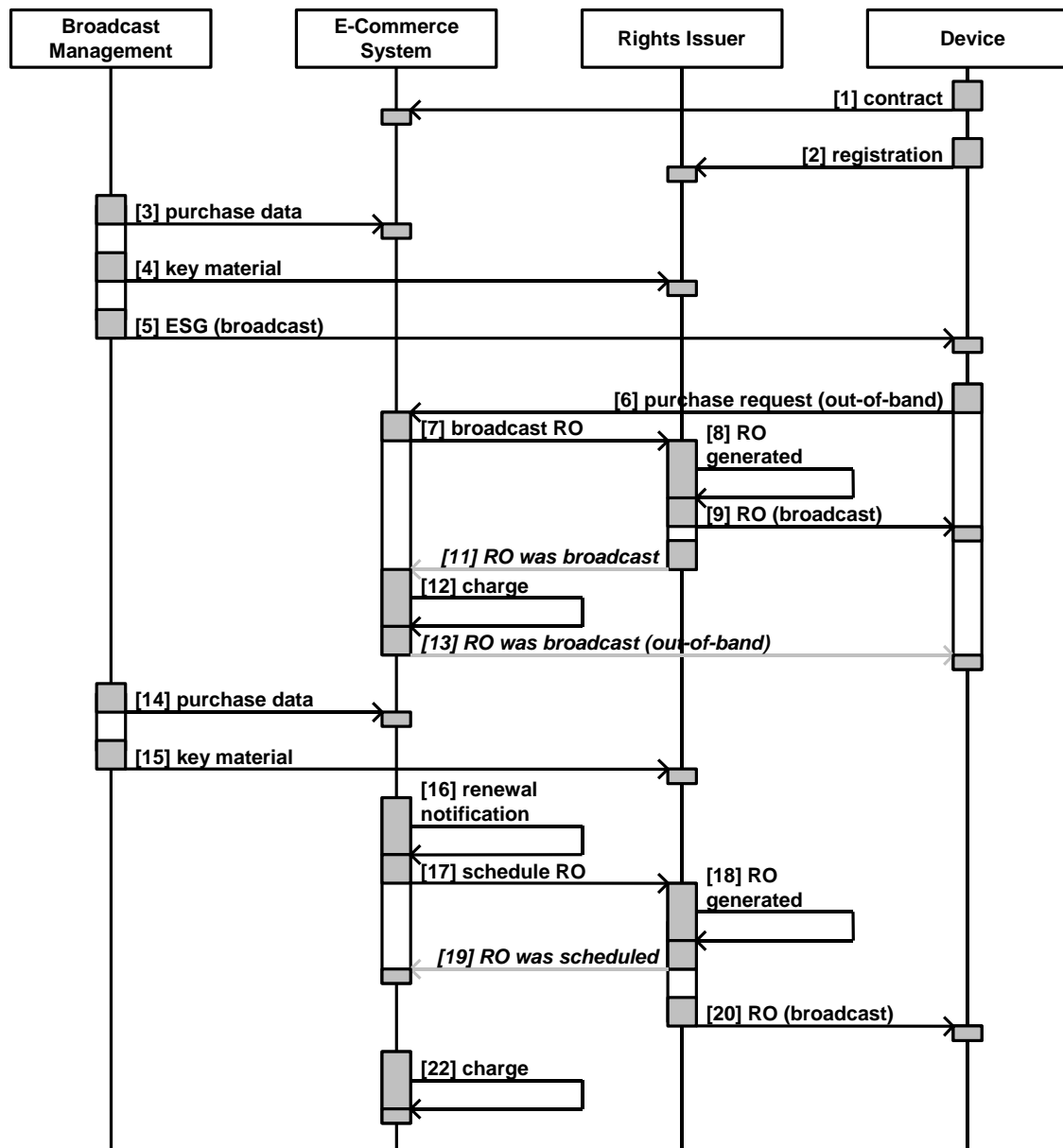


Figure 6: Purchase steps in case of a broadcast device

A broadcast device does not have the interactivity channel for communicating with E-Commerce System and Rights Issuer; hence it has to make its purchase request [6] out of band and receive its ROs [9] and [20] over the broadcast channel.

The [1] contract with the E-Commerce System could be implicit in this case, since the E-Commerce System cannot authenticate the device; instead, the user of the device is authenticated to enable the payment. Still, the device has to be [2] registered to the Rights Issuer, in order to enable the production of ROs for it. In order to reduce the bandwidth required for RO broadcasts, Broadcast Rights Objects (BCROs) can be addressed to groups of devices instead of a single device. As the BCROs can be addressed to groups they cannot be encrypted with the public key of a single device. Instead they are encrypted with symmetric keys delivered to the devices at registration. In order to reduce the bandwidth even further the BCROs do not use the XML format but instead a binary format.

As before, service data is distributed to the parties in steps [3] through [5].

Now, the user of the device [6] requests the purchase out-of-band, possibly maintaining out-of-band connection until the confirmation [13] is received. The out-of-band connection could be a phone call to the Customer Care Centre of the broadcaster; a series of HTTP requests to a web shop (charging e.g. a credit card); or in some other way. The device could be listening to the broadcast channel at this point, to immediately receive the RO broadcast at step [9]. However the RO broadcast can also be repeated in a carousel like manner giving devices the chance to receive the ROs at a later stage.

The following is one possible out-of-band purchase scenario, starting with the out-of-band purchase request [6]. The E-commerce System [7] requests the Rights Issuer to [8] generate the RO, and to [9] broadcast it immediately. There is of course no [10] response about the device receiving it (and hence no interaction [10] in Figure 6 either). However, the Rights Issuer can [11] confirm the *attempt* of broadcast to the E-Commerce System, which will [12] charge the user, and could [13] pass the confirmation out-of-band to the human user of the device. At this point, the human user can verify that the device has indeed received the RO, and possibly request re-broadcast if that was not the case.

Thus, as *one possible* implementation, “immediate broadcast” *can* be used when a new service is purchased.

Yet, when ROs are renewed, it is more convenient not to require out-of-band interaction, but to let the device receive the renewed ROs automatically. For this purpose, as a part of the broadcast channel air interface specification, delivery carousels for broadcasting ROs are specified, and a schedule for RO broadcast can be provided, allowing devices to determine the times when they listen for renewed ROs (and actually for first purchased ROs also, as an alternative to the “immediate broadcast” scenario above).

Again, RO renewal is caused by [14] purchase data and/or [15] key material being changed.

Now [16] the renewal notification will again take place in the E-Commerce System, but it is not communicated to the device; instead, the E-Commerce System [17] requests Rights Issuer to [18] generate the renewed RO and to enter it into the broadcast carousel; the E-Commerce System then receives [19] confirmation thereof. In this case, the actual [20] ROs are typically broadcast continuously in a carousel format during their period of validity, since there is no [21] response (not in the figure either) for their success or failure.

ROs may be broadcast well before the time of the actual key change, so that as the programme or service key changes, access to the service is continuous.

As in the interactive device case, [22] charging of a continuous subscription may be repeated as well, by the E-Commerce System

Should the device indeed fail to receive *any* of the broadcast attempts [20], then, as a fallback procedure, the human user of the device may again contact the E-Commerce System (represented by customer care, web shop or alike) out-of-band and request an “immediate broadcast” of the renewed RO. Thus, as one implementation option, requested “immediate broadcasts” and scheduled, repeated broadcasts using a broadcast carousel are alternative ways of achieving the same goal. From the device and air interface points of view, the essential thing is when and how to listen for the ROs.

Again the secure DRM agent of the device at all times holds an RO with the key material for decrypting the service, the RO being encrypted either by the device specific public key, or another key securely held by the DRM agent.

4.5 Consumption Steps

While the purchase steps may be applicable to both file download and streaming services, here we shall concern ourselves with streaming services only. What is achieved by this specification is that a generic data stream (not tied to any particular application) can be transmitted over a broadcast channel protected in such a way, that it can be consumed (i.e. received off air and presented to the user) by only those devices, which have purchased the proper Rights Object (RO) for the service.

To understand content consumption, we have to look at the key hierarchy which is applied for service protection.

- At the highest level, we have the keys applied by the standard DRM mechanism, used to protect the ROs in such a way that only the secure DRM agent in the correct device has access to the key material contained in each RO.
- Next, we have service keys, whose lifetime is relatively long, and which are typically used to protect continuous services such as television channels.
- Optionally, we may use programme keys, whose lifetime covers only a specific part of the service, such as a particular television programme. Programme keys are required in the case where rights associated with a service and indicated in a key stream message change frequently, with each programme event.
- At the lowest level, we have traffic keys, whose lifetime is relatively short, and which are used to encrypt the actual data stream.

To buy a pay-per-view RO for a particular television programme, the device purchases an RO containing a programme key, which decrypts traffic keys, which in turn decrypts the data stream.

To buy a television channel RO (for some time duration), the device purchases an RO containing a service key, which decrypts programme or traffic keys (where the programme keys, if used, decrypt the traffic keys), while the traffic keys decrypt the data stream. Let us look into this latter case in more detail, in case of one service. Our key hierarchy is then applied in the following fashion:

- The purchased RO contains a service key (which only the secure DRM agent of the device can reveal from the RO). As a consequence of the purchase steps, the DRM agent already holds the RO.
- Programme/traffic keys are broadcast as a separate key stream, encrypted by the service key. This is similar to the broadcast of OMA DRM 2.0 DCFs in the sense that by using the RO, the DRM agent can decrypt the programme/traffic keys from the key stream messages (but the key stream message format is different from DCF format).
- The data stream is then broadcast encrypted with the traffic keys. (If programme keys are present in the key stream, they decrypt the traffic keys.)

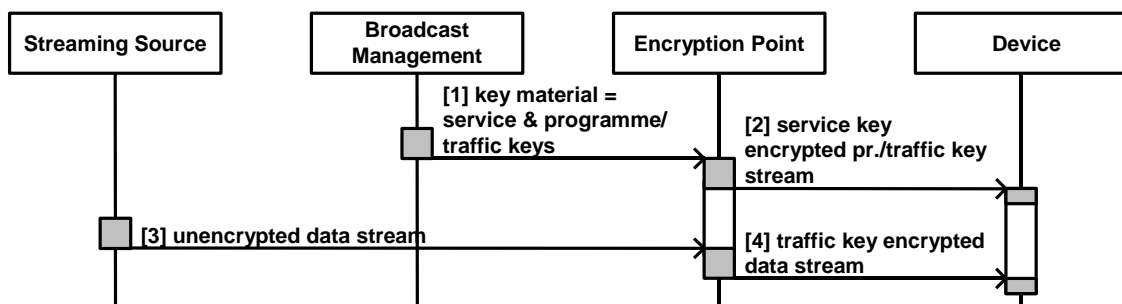


Figure 7: Consumption steps from the broadcaster point of view

Figure 7 illustrates the broadcast steps. Broadcast management [1] provides the key material consisting of service and programme/traffic keys to an encryption point, (an abstract network function) which may reside at alternative places in the network. As the programme/traffic keys are to be used for data stream encryption, the encryption point [2] broadcasts the programme/traffic keys, encrypted with the service key, to devices as key stream messages. The [3] unencrypted data stream is then [4] broadcast to devices encrypted with the traffic keys.

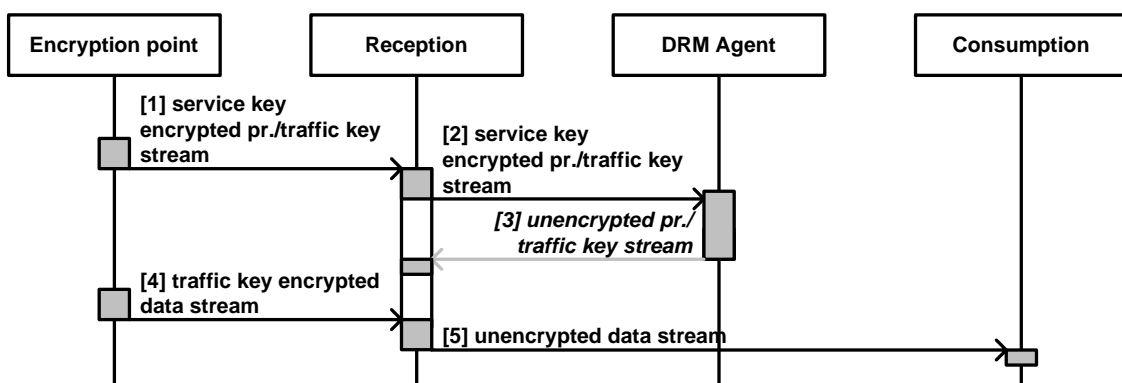


Figure 8: Consumption steps from the device point of view

Figure 6 then illustrates consumption steps at the device, in a generic way.

The [1] key stream messages are received at some reception function in the device and [2] passed to the DRM agent. The DRM agent then internally reveals the service key from the RO it holds; decrypts the programme/traffic keys from the key stream messages and [3] returns the traffic keys in plaintext to the reception function.

The reception function then [4] receives the encrypted data stream, decrypts it with the traffic keys and [5] passes the plaintext data stream to the consumption function of the device, such as a television display screen. (Yet the data stream may be anything, not just television programmes – the service protection solution specified herein is independent of the data stream format and purpose.)

Various realizations of the abstract Figure 8 are possible, and actually the data stream decryption can reside at two alternative protocol levels, both of which have to be supported by devices:

- Data stream decryption can be applied at IPsec level, which naturally lends itself to an implementation where the reception process (with data stream decryption) is part of the IP stack of the device. In this case the application may be completely independent of the solution: it need not know anything about the IPDC service protection.
- Alternatively, data stream decryption can be applied at SRTP level, which lends itself to an implementation where the reception process (with data stream decryption) is integrated with the application itself. A choice between IPsec and SRTP is determined by various factors; application independence versus integration being only one of them.

4.6 Service Protection vs. Content Protection

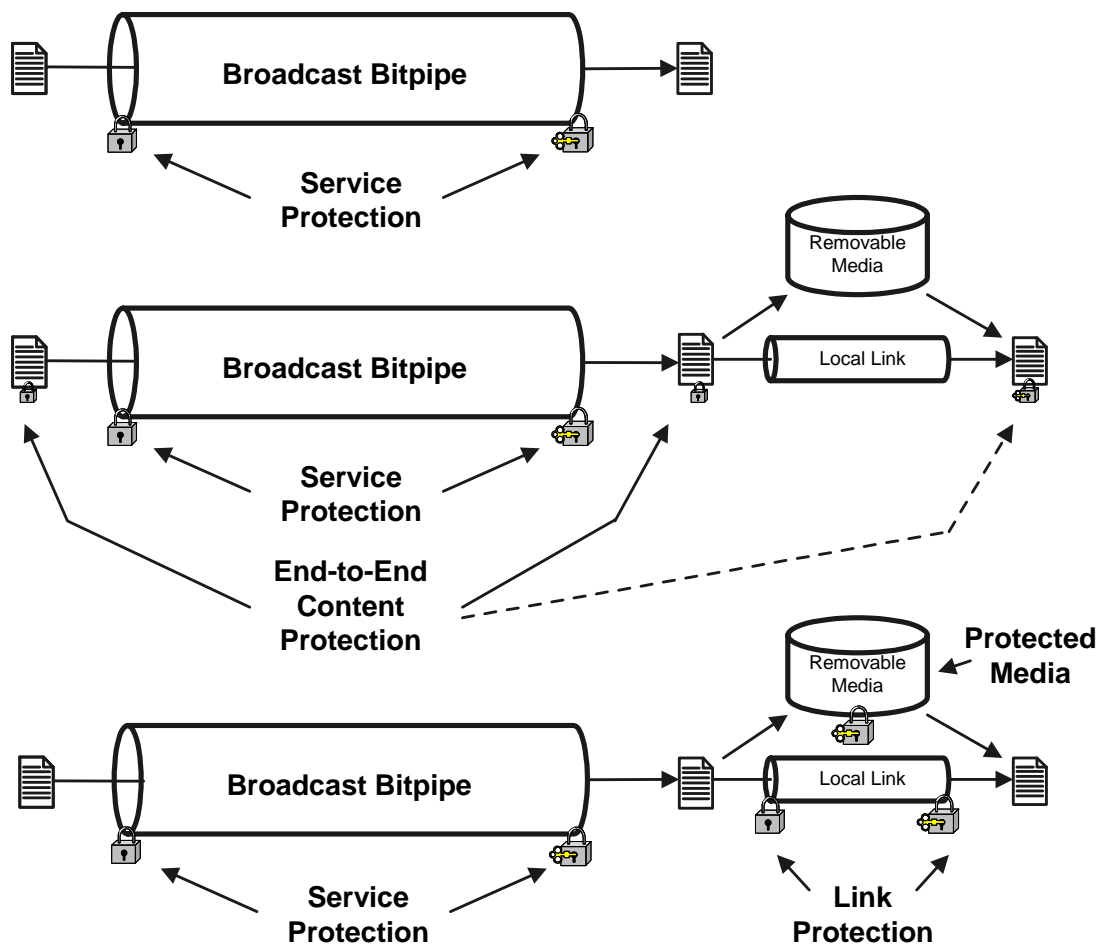


Figure 9: Service protection vs. content protection

Figure 9 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 IPDC Services Purchase and Protection system defined in the present document 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, the play permission is required. If only service protection is needed, the 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, such as DVB-CPCM. The Usage State Information needed by such a content protection system can be carried as a constraint for the export permission.

5 Theory of Operation

5.1 End-to-end Architecture

The figures in this section can be used as a reference and overview when reading this specification. It contains a network-centric end-to-end architecture and a figure that shows how the Public Key Infrastructure relates to the end-to-end architecture.

The end-to-end architecture maps the elements that this specification refers to onto the entities defined in the tentative IPDC in DVB-H and OMA-BCAST architectures. [Note to the editor: to be updated when the architecture specifications are finalised.]

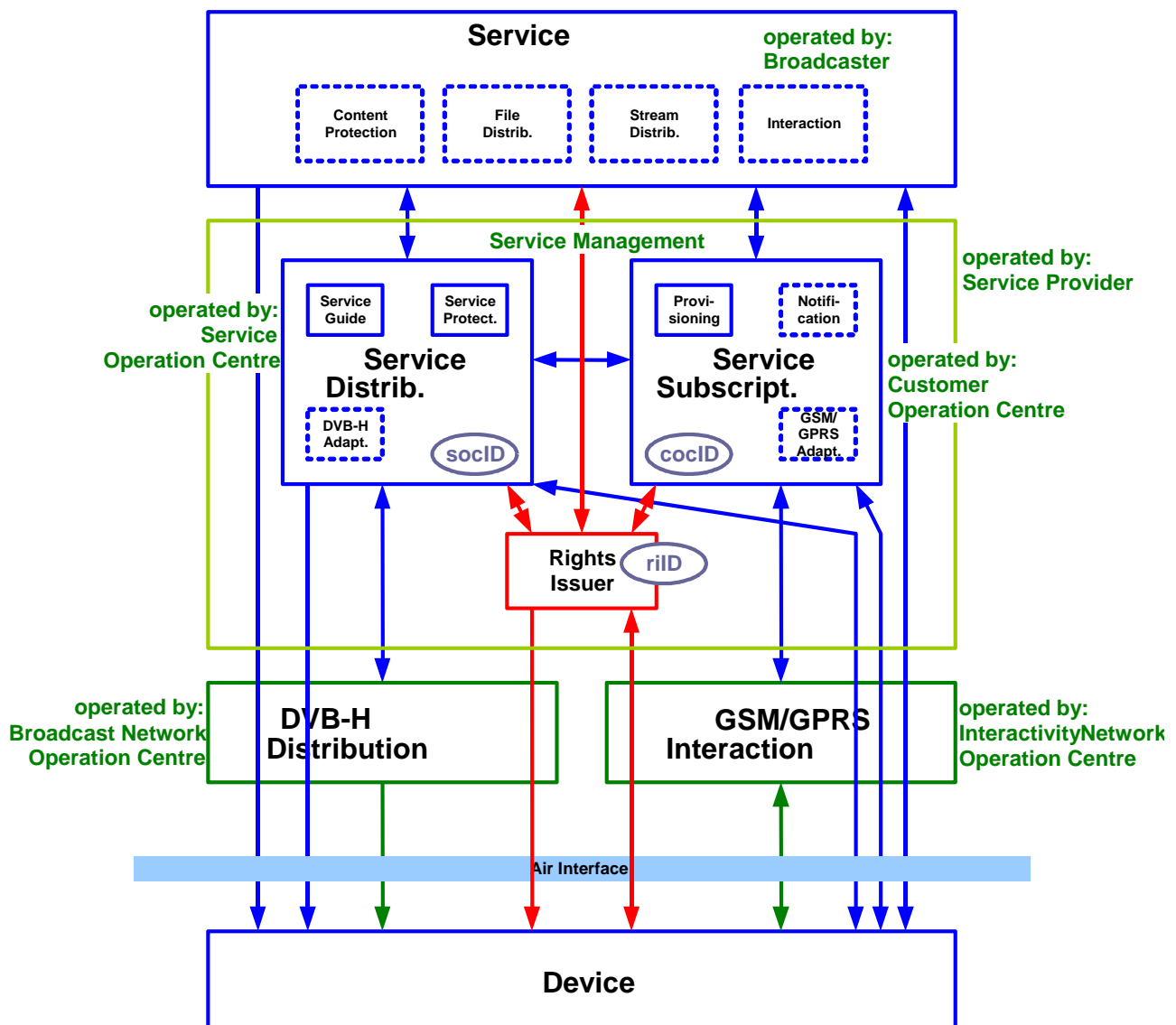


Figure 10: Service Protection and Purchase Entities and names (Broadcast Architecture)

Figure 10 above names several entities used in this specification. It also maps the entities used in this specification to the related architectural entities in the tentative IPDC in DVB-H architecture and OMA-BCAST Architecture. [Note to the editor: This has to be checked once the relevant specifications are available.] 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 separate entities here may well be merged into larger systems. One should also note that there are

some missing 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 described in this specification. Service Subscription Management takes care of the purchase of services. The Rights Issuer takes care of device registration and rights issuing.

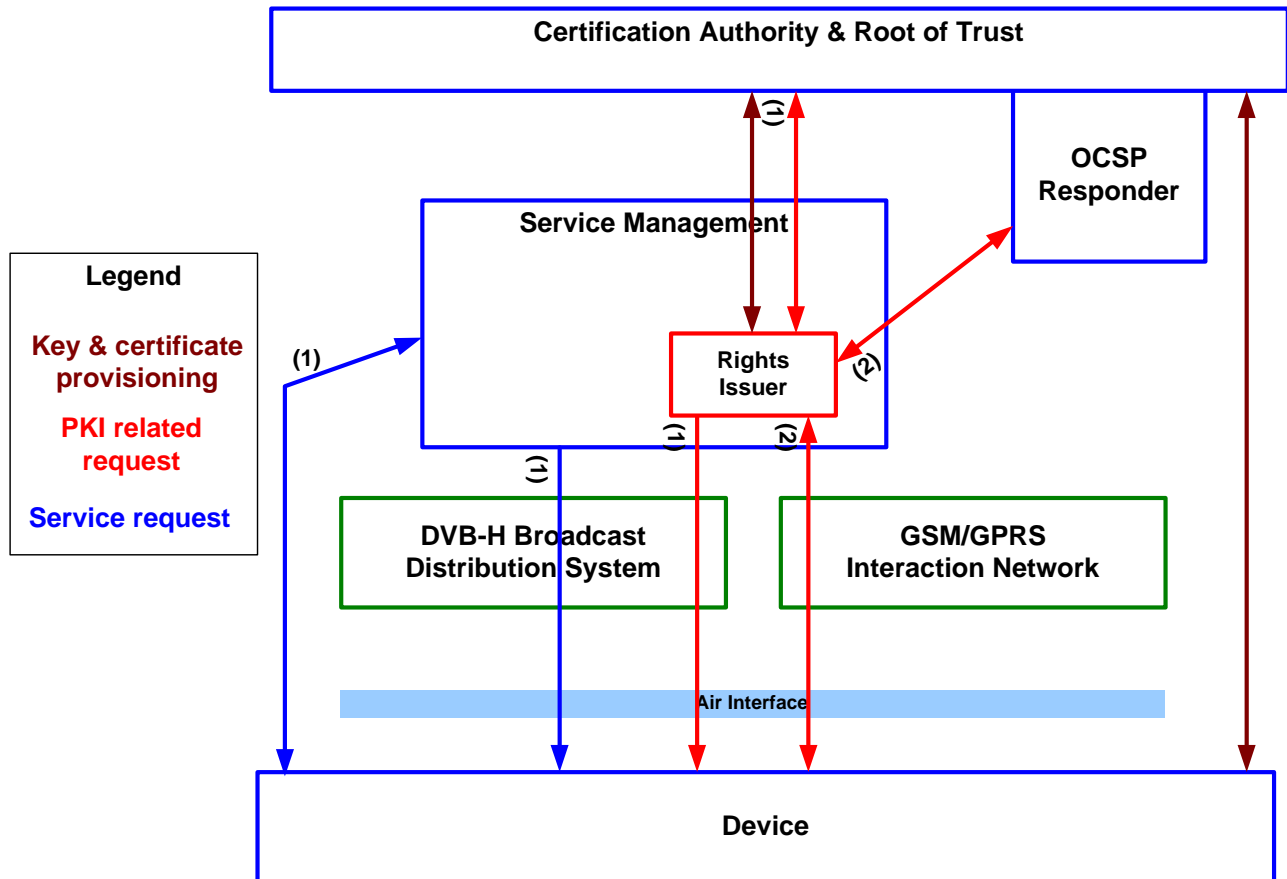


Figure 11: Public Key Infrastructure

Figure 11 above maps entities from the Public Key Infrastructure to the broadcast architecture. A Root Certification Authority (CA) provides keys and certificates to devices and Rights Issuers. The device and Rights Issuer SHALL have the same Root CA, otherwise the Rights Issuer cannot register the device or issue rights to the device.

Lines marked with (1) relate to broadcast mode devices and lines marked with (2) relate to interactive mode devices (please compare figure 11 with figures 13 and 14).

5.1.2 Special Cases

5.1.2.1 Free-To-Air Services

Free-To-Air services are broadcast without any service protection. Any device can receive these services.

- Free-To-Air services SHALL NOT be broadcast within IPsec or SRTP. 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 specification.

5.1.2.2 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”.
- Broadcasting Rights Objects to all devices in a population or to particular pre-existing Broadcast 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 specification.

5.2 Electronic Service Guide and Purchase

[Note to the editor: This chapter sets out some requirements on the operation of the ESG. These must be checked once the ESG specification is complete and available.]

The Electronic Service Guide is available to all DVB-H devices. It is a source of service and programme information. Service discovery and purchase of services is based on information transmitted in the ESG. For each service and programme, the ESG contains all the necessary information for making a purchase and for the device to find services and programmes.

Figure 12 illustrates a high-level overview of an ESG and purchase system. Line (1) in Figure 12 represents the broadcast of an ESG over DVB-H. Lines (4) and (5) represent purchase and lines (2) and (3) represent the delivery of necessary Rights Objects.

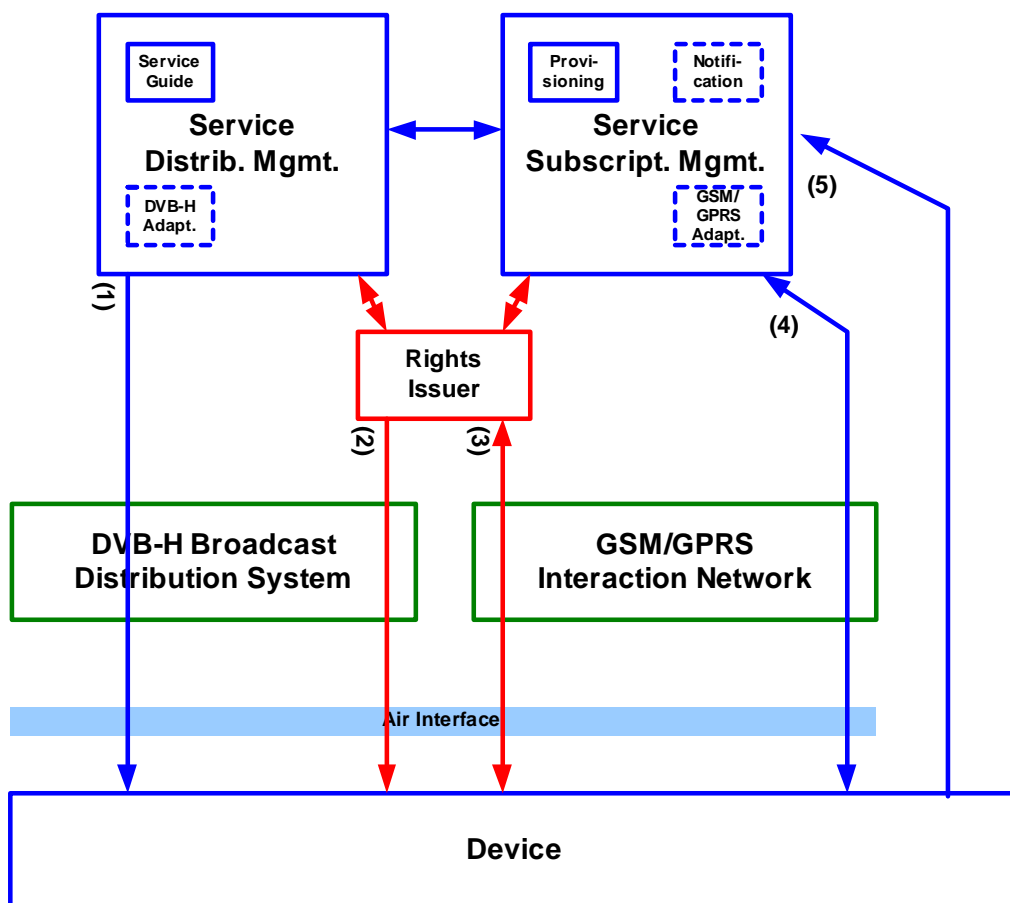


Figure 12: Overview of ESG and Purchase

ESG data is carried over the broadcast channel. To be able to use the protected services described in this specification, a device needs to be able to receive and parse ESG data. Unless the service is free-to-air (see 5.1.2.1) the user needs to purchase the service. The ESG contains all the needed information to make the purchase. Interactive and mixed-mode devices can find the URI and other necessary parameters from the ESG. For broadcast mode devices, the ESG contains

a ServiceID (see B.1.3.2) that identifies a purchased service. Line (4) in figure 12 represents purchase by interactive and mixed-mode devices over the interactivity channel and line (5) represents off line purchase by broadcast devices.

With interactive and mixed-mode devices, purchase initiates the sending of an ICRO over the interactivity channel; line (3) in figure 12. With broadcast and mixed-mode devices, BCROs are sent over the broadcast channel. Before purchases can happen, devices need to be registered with the relevant Rights Issuer – see chapter 5.3.

5.3 Registration

5.3.1 Concept of the 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 (a.k.a. PKI) and a Certificate Revocation List (a.k.a. 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.

In the case that the device supports an interactivity channel which can be used to contact an RI, the device is called an “interactive device”. In case 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 as:

Table 1: Registration types

registration type	device types	section
register device for interactive mode of operation.	Interactive	5.3.2
register device for broadcast mode of operation. (content & RO via broadcast channel)	Broadcast and Mixed-mode	5.3.3
register device for mixed-mode of operation. (both interactive and broadcast)	Mixed-mode	5.3.4

On successful execution the registration operations result in the creation of the 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.

5.3.2 Registration for interactive mode of operation

Registration for interactive mode of operation when OMA DRM 2.0 is used on Registration Layer is according to [OMA-DRM-DRM], using the standard 4-Pass ROAP.

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

5.3.3 Registration for broadcast (only) mode of operation

To register 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 the interactivity channel is not able to make a connection to the RI, but the device is able to create an other connection to a connected OMA device. This device is called an unconnected interactive/unconnected mixed mode device, and is covered in 5.3.2.

- The device has no interactivity channel and is unable to make a connection to an interactive device. This device is called a broadcast (only) device. In this case the 1-pass binary push registered device protocol is used, as is specified in this document.

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 document.

Following sequence chart explains the registration for broadcast only mode of operation.

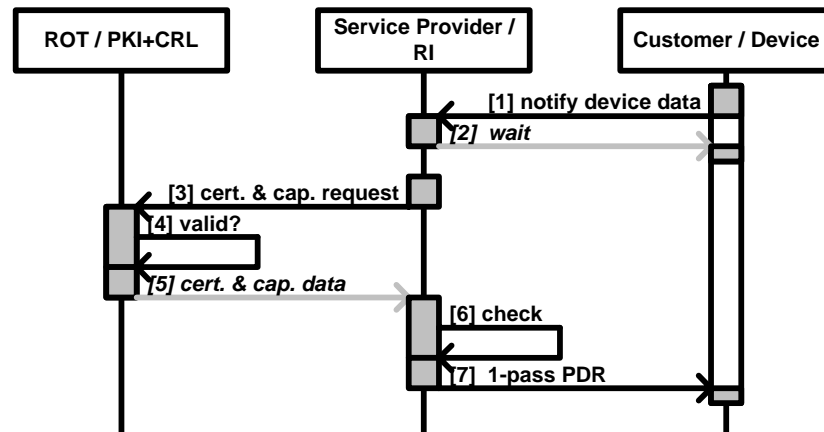


Figure 13: Registration for broadcast mode of operation with one ROT

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 the broadcast channel.

Explanation of the protocol:

- Once the RI has the device data from the device [1] via the protocol described in section 6.4.3.2, 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 (a.k.a. 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 described in section 6.4.3.4. The registration data (in the format of the `device_registration_response()` message) is specified in section 6.4.3.7.2. 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.

Subsequent distribution of Right Objects at regular intervals is done with a message send as an inform message using the 1-pass Inform Registered Device protocol.

5.3.4 Mixed-mode registration for interactive and broadcast modes of operation

This section applies for devices supporting both communication via a broadcast channel and an interactivity channel. If such devices are registered for both interactive and broadcast mode of operation, the RI has the option of sending ROs either over the Broadcast Channel or the 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 section 5.3.3), 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 14 . Steps 1-8 in this figure are the 4-pass ROAP as specified in [OMA-DRM-DRM] with which the registration for interactive mode of operation is done when OMA DRM 2.0 is used on Registration Layer.

After step 8, a device that is capable of broadcast operation and uses OMA DRM 2.0 for registration 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 9 in the figure). 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, when OMA DRM 2.0 is used to initiate the registration, 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 8.1.3.5. From this capabilities data, the RI finds out that the device is also capable of broadcast mode of operation. In such case the RI MAY decide to send this device the device_registration_response() message (step 14 in the figure), 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 section 6.4.3.6, 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. Refer to section 8.1.3.5 for details on the use of this XML structure.

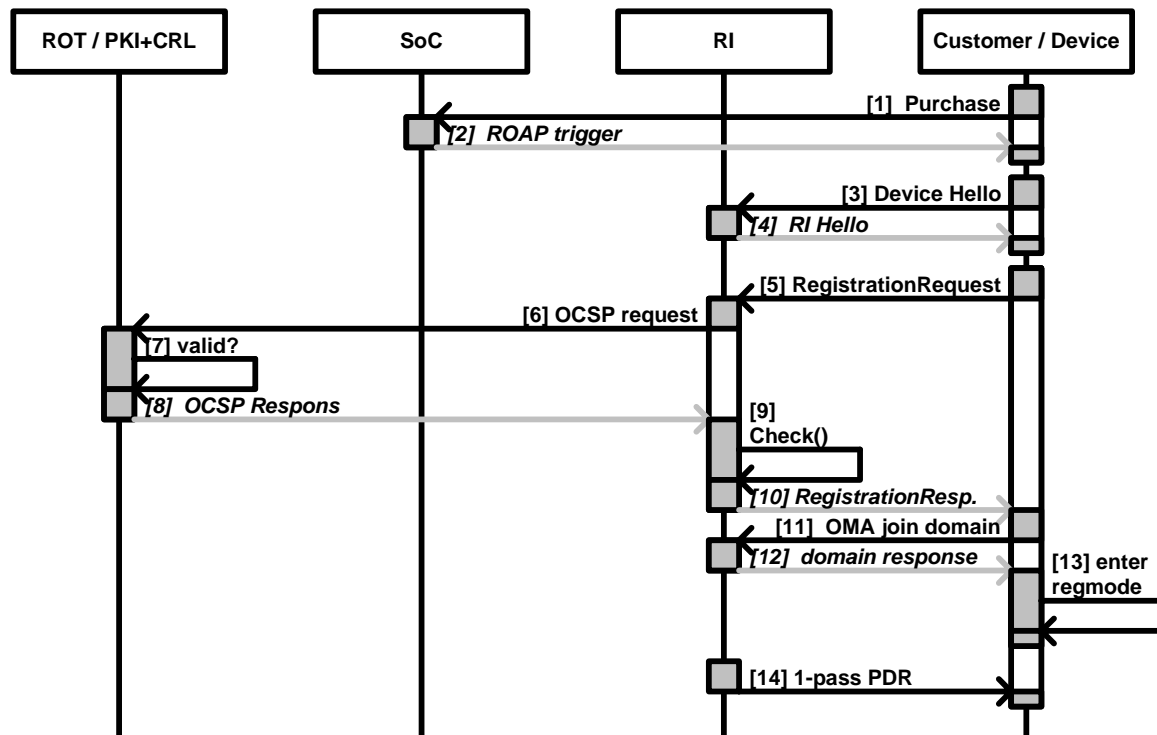


Figure 14: Registration for mixed-mode operation with one ROT

5.4 The Four Layer Model

5.4.1 Key Hierarchy

A four-layer key hierarchy is used to implement service protection.

5.4.1.1 Keys on the Traffic Layer

On the lowest level, the data is encrypted using either IPsec or SRTP. 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 key 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.

5.4.1.2 Keys on the Key Stream Layer

The TEK itself is transmitted on the key stream layer. The TEK will 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 described in the following three sections to be used.

5.4.1.2.1 Service based subscription

If the service is made available to customers by subscription only, then:

- 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 described in section 5.4.1.2.2 is used.
- If access rights do not change per programme, a programme key is not used and the scheme below is 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 can be normal OMA DRM 2.0 ROs in the case of an interactive device or BCROs for both mixed-mode and broadcast only devices.

Figure 15 shows the key hierarchy for the case of a service based subscription.

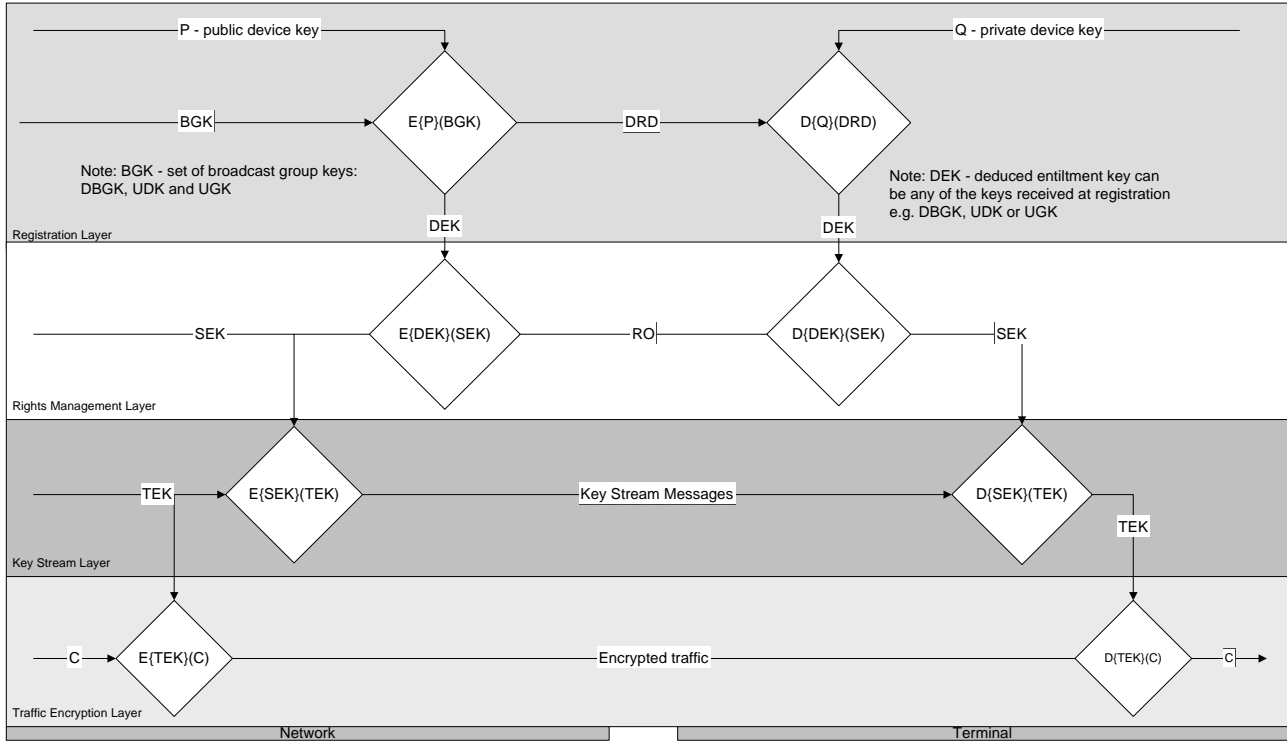


Figure 15: 4-layer key hierarchy - use of SEK only

5.4.1.2.2 Pay-per view based and service based subscription

If content is made available both via a service subscription and via a pay-per view based subscription then the TEK will 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 will 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. So 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.

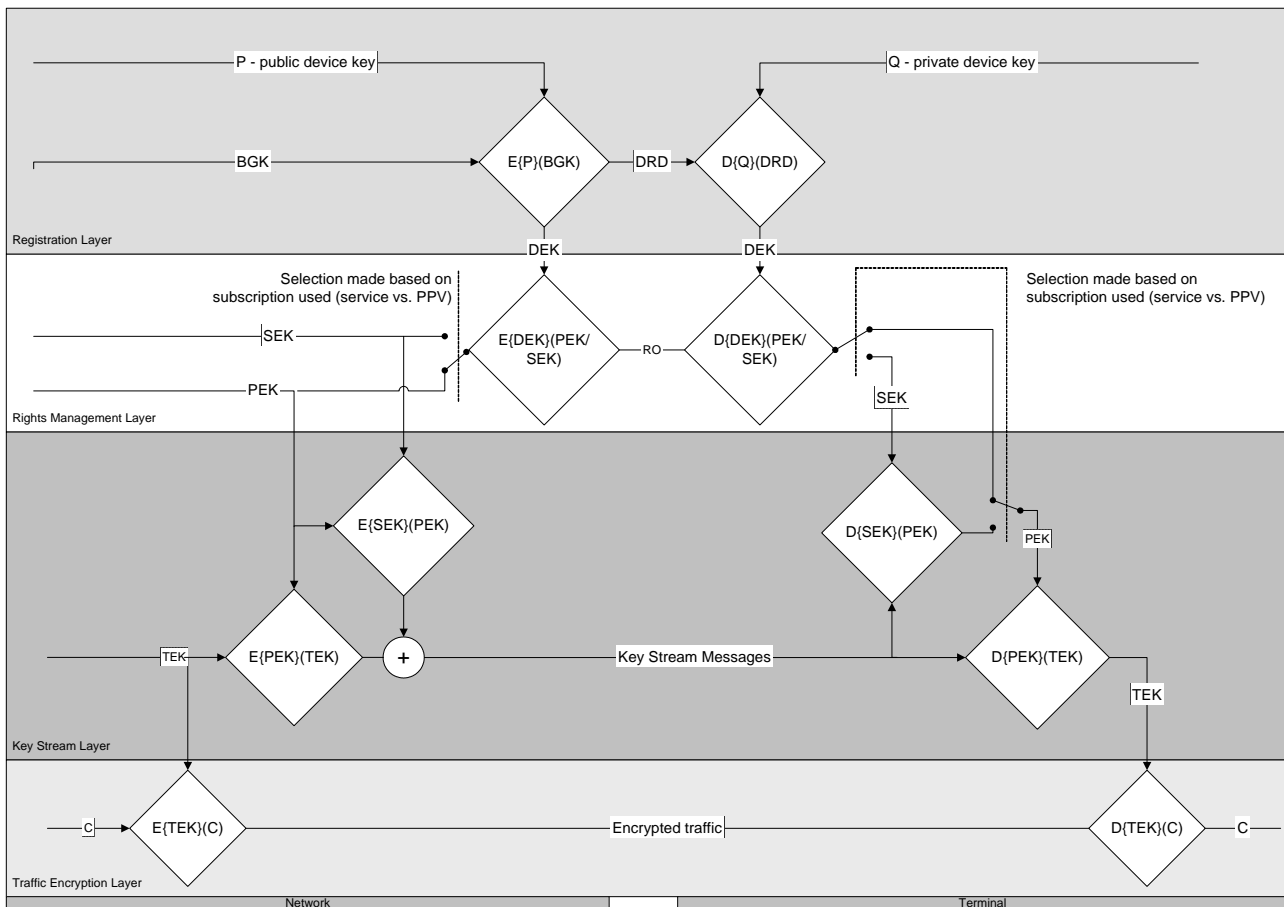


Figure 16: 4-layer key hierarchy - use of PEK and SEK

Figure 16 shows the four layer key hierarchy in the case of service subscription and pay-per-view.

5.4.1.2.3 Pay-per view based consumption

If content is consumed on a pay-per view only basis, that means that the content is not available via a service subscription, than 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.

5.4.1.3 Keys on the Rights Management Layer (Broadcast mode)

The SEK and PEK are transmitted to the device on the Rights Management Layer as part of a BCRO.

The keys used to encrypt and decrypt the SEK or PEK depend on the addressing mode of the BCRO (see section 5.6.2). The term “Deduced Encryption Key” (DEK) is used to describe 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 that an RO is addressed to a unique device, the DEK used to encrypt the SEK or PEK is the unique device key (UDK) which was delivered during device registration.
- RO addressed to a broadcast group (subset of unique group)**
 In the case that an RO is addressed to a subset of a unique group (broadcast group), the DEK is deduced from the broadcast group keys (BGKs) by use of zero message broadcast encryption. Refer to section 5.6.3 for an explanation of the zero message broadcast encryption concept. Refer to section A.8 for an explanation of how the DEK is deduced given the bit access mask and the BGKs.

- **RO addressed to a unique group:**

In the case that an RO is addressed to all devices in a unique group, the DEK used to encrypt the SEK or PEK is the unique group key (UGK).

- **RO addressed to a domain:**

In the case that an RO is addressed to a domain, the DEK used to encrypt the SEK or PEK is the local domain key (LDK) which was delivered during device registration.

- **RO containing a CEK:**

In the case an RO is for an OMA DRM 2.0 content format (e.g. a DCF), the asset carries a CEK object and an additional cipher value. Decryption of the key material is defined by [OMA-DRM-DRM].

5.4.1.4 Keys on the Rights Management Layer (Interactive mode)

If a Rights Object containing the SEK or PEK is delivered via the interactivity channel, the UDK is used to decrypt the SEK/PEK. If the RO contains a CEK, it will be processed according to [OMA-DRM-DRM] and is beyond the scope of this specification.

5.4.1.5 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	
BGK	broadcast_group_key	used for zero message broadcast
LDK	local_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
SLDF	shortform_local_domain_filter	domain_id address, not a key
LLDF	longform_local_domain_filter	domain_id address, not a key
TDK	token_delivery_key	not used on registration layer

The keyset itself is delivered to the device in a protected format as part of the device registration data (refer to section 6.4.3.7.2 for details).

The RI generates a session key (SK) to protect the keyset_block (UGK, BGK1..n, UDK, LDK, RIAK, UDF, SLDF, LLDF and/or TDK), which carries the keyset described in table 2 above.

$$encrypted_keyset_block = E\{SK\}(keyset_block)$$

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

$$sessionkey_block = E\{DP\}(SK + encrypted_keyset_block)$$

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

Note: If the keyset_block would not fit into the size of the sessionkey_block the remainder is kept as surplus_block. Refer to section 6.4.3.7.2.2 for details.

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

$$signature_block = A\{RIQ\}(message)$$

where the RIQ is the private key of the RI.

Upon reception the device follows the rules described above in reverse order:

$$V\{RIP\}(signature_block)$$

$$SK + encrypted_keyset_block = D\{DQ\}(sessionkey_block)$$

$$keyset_block = D\{SK\}(encrypted_keyset_block)$$

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 device's private key (DQ).

Note: If the surplus_block is present, it is concatenated to the keyset_block from the sessionkey_block. Refer to section 6.4.3.7.2.2 for details.

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

The notation HMAC_SHA1{k}(s) is used to denote the computation of HMAC [RFC 2104] with SHA1 [FIPS 180.2] 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 DEK is "derived" from the UGK, BGK, UDK or LDK to decrypt the BCRO, such that

$$DEK = HMAC_SHA1_128\{UGK\}(salt)$$

or

$$DEK = HMAC_SHA1_128\{NK_i || \dots || NK_j\}(salt)$$

where the NK's 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 described in section 5.6.3. See Sections 5.6.3 and A.8.

or

$$DEK = HMAC_SHA1_128\{UDK\}(salt)$$

or

$$DEK = HMAC_SHA1_128\{LDK\}(salt)$$

The DEK is used to decrypt the part of the BCRO containing keys, which carries CEK or SEK and/or PEK. 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.

5.4.1.6 Authentication overview

Following picture explains the authentication "hierarchy" of the system.

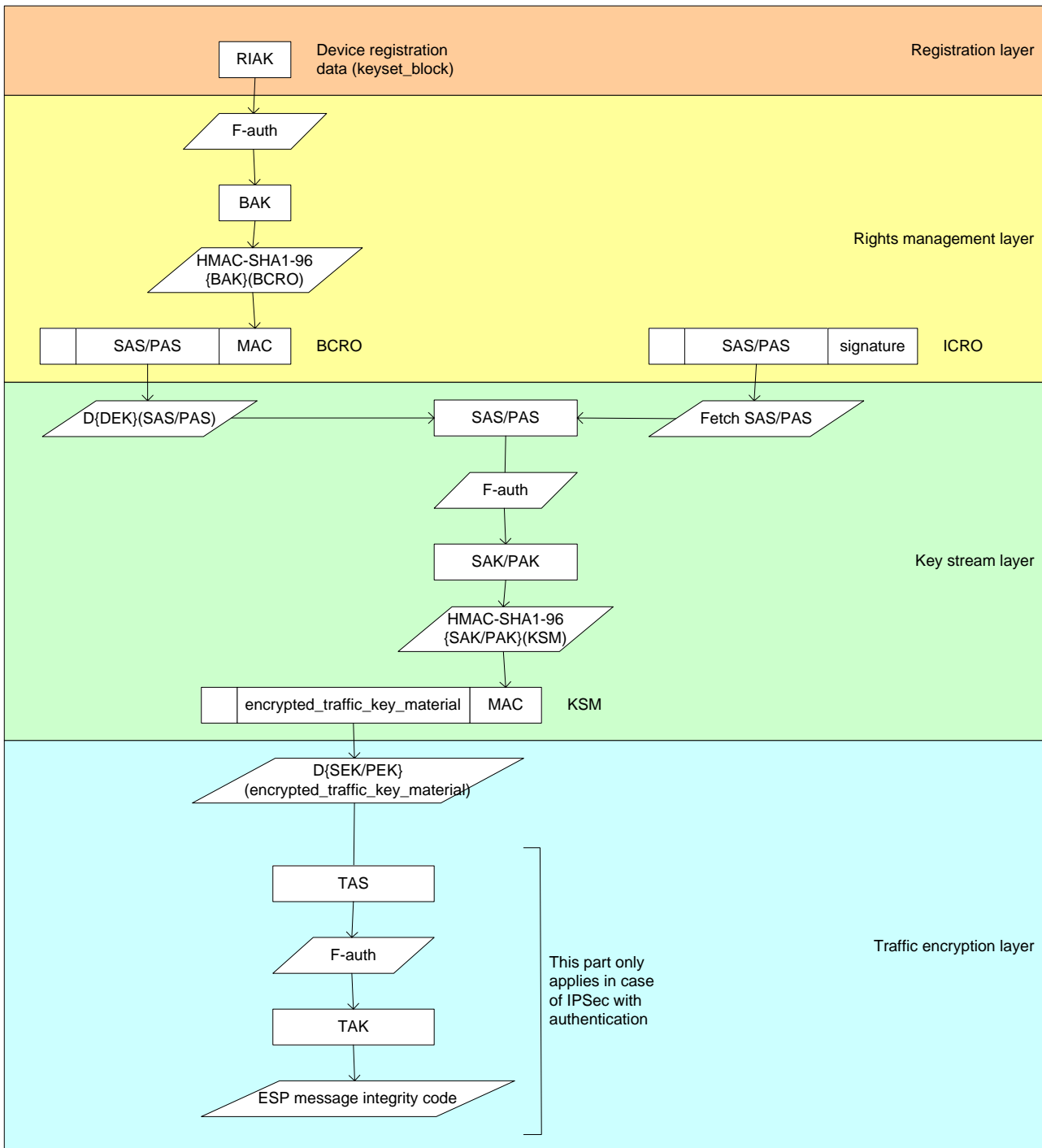


Figure 17: Authentication hierarchy

5.4.1.6.1 Authentication keys on 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 BCRO Authentication Key (BAK) which is derived from the RI Authentication Key (RIAK) which is delivered during registration (see section A.9.1).

5.4.1.6.2 Authentication keys on 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 Program Authentication Key (PAK) and/or the service authentication key (SAK), which are derived from the Program Authentication Seed (PAS) and the Service Authentication Seed (SAS), which are delivered as part of the RO (see Annex A.9.2)

5.4.1.6.3 Authentication keys on 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 section A.9.3).

5.4.1.6.4 Authentication keys on registration layer (broadcast mode)

The RI Authentication Key (RIAK) is delivered during registration as part of the `device_registration_response()`. Refer to section 6.4.3.4 for details.

5.5 Deployment for interactive mode of operation

5.5.1 Concept of Domains – OMA DRM 2.0 Domains

Content in OMA DRM 2.0 can be bound to a device or to a domain – see [OMA-DRM-DRM].

A domain in OMA DRM 2.0 is a group of one or more devices which share a common secret, the so called Domain Key. In the case of content that is bound to a domain, the Content Encryption Key for that content (as stored in the RO associated with that content) is encrypted with the Domain Key of that domain. A device which 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 DRM 2.0 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 needs to have joined the corresponding domain.

In this specification, an OMA DRM 2.0 domain as specified by [OMA-DRM-DRM] is referred to as an interactive domain.

5.6 Deployment for broadcast mode of operation

5.6.1 Concept of Domains – local domains

Content in OMA DRM 2.0 can be bound to a device or to a domain, see [OMA-DRM-DRM]. In this specification, we refer to a domain as specified by [OMA-DRM-DRM] as an interactive domain.

A domain in [OMA-DRM-DRM] is a group of one or more devices which 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 which 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-DRM-DRM] 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 has to have joined the corresponding domain. The equivalent for these protocols in the case that there is no interactivity channel are defined in section 6.4.4.

The equivalent of the interactive domain in case there is no interactivity channel is the local domain. In the BCRO, there is a facility to indicate that the BCRO is intended for a local 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`.

Please note that while the domain ID and domain key MAY be the same in the case of the interactive and local domain, the content for local domains with BCROs is not interoperable with content for interactive domains with ICROs.

5.6.2 Addressing (group / subset / device / domain)

The registration data supports four methods of addressing devices, as is explained in figure 18 below.

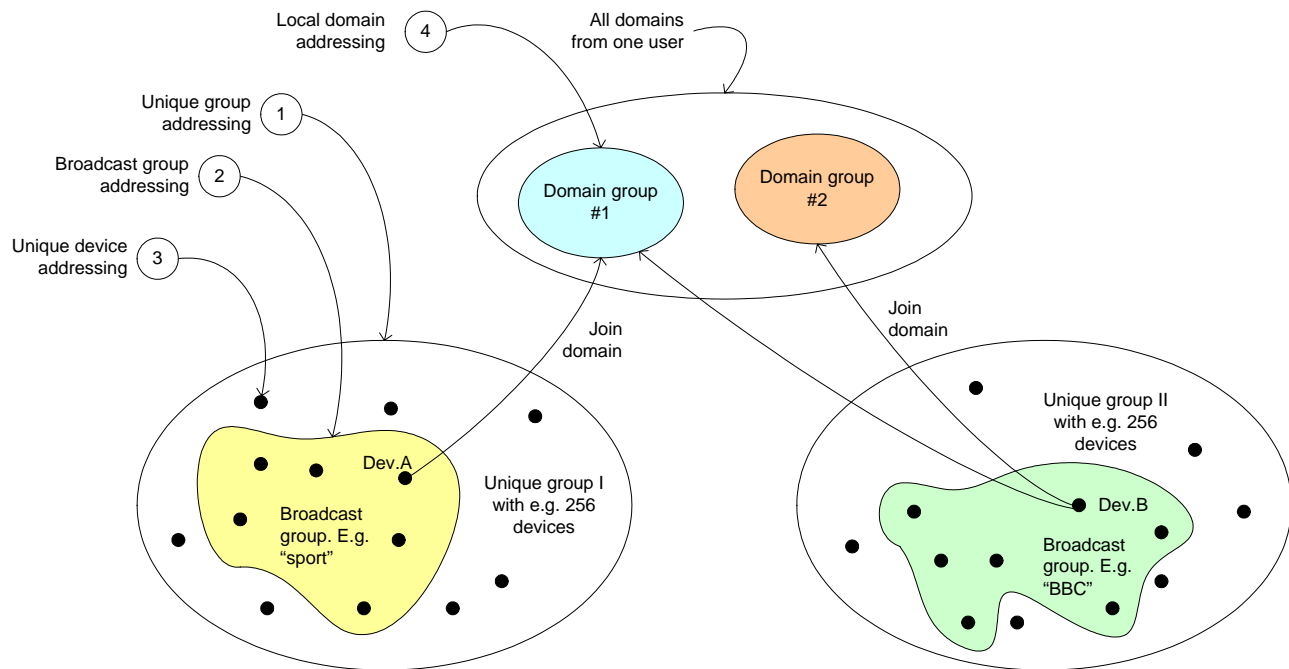


Figure 18: Explaining the concept of addressing

key:

- 1 = addressing a unique group – use Unique Group Key (UGK)
- 2 = addressing < unique group – use Broadcast Group Key(s) (BGK)
- 3 = addressing only one device – use Unique Device Key (UDK)
- 4 = addressing a local domain – use Local Domain Key (LDK)

A unique group contains all the devices in a group. A broadcast group can be smaller than, or as large as, the unique group. Alternatively a device can be addressed via a (local) domain group. A unique device can be part of a unique group and/or a broadcast group and/or a local 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 to many devices.

The following sections describe 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 content key is encrypted with a Deduced Encryption Key (DEK) by the RI. The following sections describe the different addressing modes, how the message is filtered and what type of DEK 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 BGK and/or UDK and/or LDK to the device at registration time.

Note: Refer to section 5.4.1 for the general overview of the process used to construct the DEK.

5.6.2.1 Addressing the unique group

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

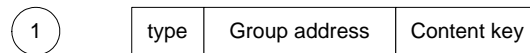


Figure 19: (Oversimplified) group BCRO

- The group address was delivered with the registration data and is used to filter for the message (refer to 6.4.3.7 for details).
- The content key (which can carry the SEK and/or PEK) is encrypted with a Deduced Entitlement Key (DEK). In this case the unique_group_key (UGK) is used as the DEK. The UGK used by the RI is identical to the key that was delivered with the registration data sent to the device (refer to 6.4.3.7 for details).
- All the devices in the group can use the content key.

5.6.2.2 Addressing a broadcast group

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

- At registration, the registration data delivers a set of broadcast_group_keys (BGK) to the device. By using zero message broadcast encryption (refer to section 5.6.3) a Deduced Entitlement Key (DEK) can be constructed from the BGKs sent to the device. The RI uses the this DEK to encrypt the content key and only the devices with the matching set of BGKs on board can construct the same DEK.
- For reasons of completeness it is mentioned that is also possible to deliver no BGKs with the registration data. With 0 (zero) BGKs, it is of course not possible to deduce a DEK. In this case, unique device addressing with a unique_device_key (UDK) as the DEK is used. Please refer to section 5.6.2.3 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. Please refer to section B.3 for more details.

To access the broadcast group, the following (oversimplified) BCRO is used.

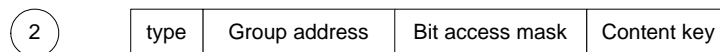


Figure 20: (Oversimplified) broadcast group BCRO

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 (refer to 6.4.3.7 for details).
- The Eurocrypt-style bit access mask 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 broadcast group can deduce the broadcast key from the zero message broadcast keys in the device.
- The device can construct the Deduced Encryption Key (DEK) from the BGKs that were delivered to the device. The BGKs used in this process belong to the same RI context as the BCRO.
- Only members of the broadcast group can use the content key. If a member of the group succeeds in constructing the DEK, that member can decrypt the Content key. Any group member that tries to deduce the DEK but does not have the appropriate (zero message) BGK on board is unable to deduce the DEK to decrypt the content key.

5.6.2.3 Addressing a unique device

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

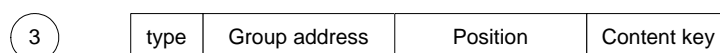


Figure 21: (Oversimplified) unique device BCRO

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 (refer to 6.4.3.7 for details).
- The content key (which can carry the SEK and/or PEK) is encrypted with a Deduced Encryption Key (DEK). In this case the Unique_Device_Key (UDK) is used as the DEK. The UDK used by the RI is identical to the key that was delivered with the registration data sent to the device (refer to 6.4.3.7 for details).
- Only the unique device which is addressed can use the content key.

Note: In all cases, the head end infrastructure composes the DEK used to encrypt the content key and determines the access mask on the basis of the created key. For the broadcast group case the head end infrastructure creates the access mask based on the corresponding (zero message) BGKs. A “type” field inside the BCRO SHALL indicate which addressing case is covered.

5.6.2.4 Addressing a local domain

To access a local domain, the following (oversimplified) BCRO is used.

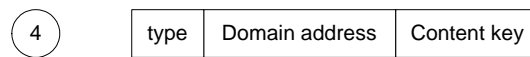


Figure 22: (Oversimplified) local domain BCRO

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

- The domain address was delivered with the registration data (Refer to 6.4.3.7 for details) and/or via a join domain response (refer to 6.4.4.1.1 for details) 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 a Deduced Entitlement Key (DEK). In this case the local_domain_key (LDK) is used as the DEK. The LDK used by the RI is identical to the key that was delivered with the registration data sent to the device (refer to 6.4.3.7 for details).
- All the devices in a local domain group can use the content key.

Note: Local domain addressing is included in the specification 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 local domain group. The "savings" in bandwidth with domain addressing are not as high as under broadcast 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.

5.6.3 Zero Message Broadcast Encryption scheme

Zero message broadcast encryption was originally introduced in [FIAT_NAOR]. 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 broadcast_group_keys (BGK) 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 specification uses an algorithm similar to the Fiat-Naor scheme, but with the following advantages:

- The algorithm is altered to support changing the DEK independently of the broadcast group keys.
- The DEK does not reveal information about the broadcast 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 ($n = \log_2(\text{group size})$). Each terminal needs (worst case) to derive $m - 1$ keys (with $m = \text{group size}$). 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:

(Note: Key material for the authentication of messages is omitted for readability reasons)

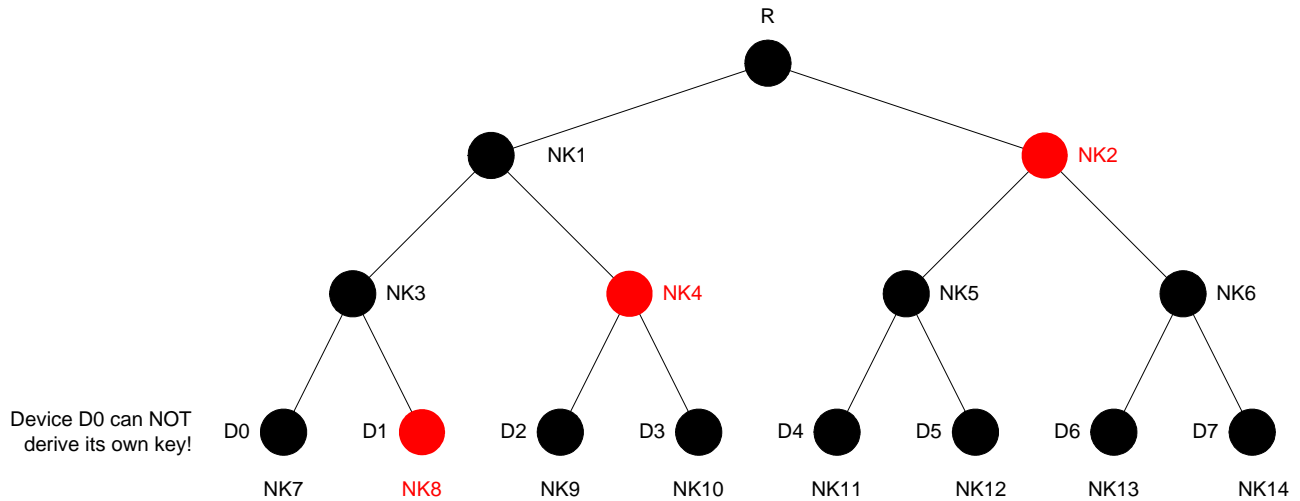


Figure 23: Example of a zero message tree with 3 nodes (keys)

$m = 8$, the number of terminal keys will be $\log_2(8) = 3$

Assume the Terminal D0 has the following keys:

$\{NK2, NK4, NK8\}$

The terminal now derives, as specified in appendix A.8 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 $AES_K(T1+2i)$ and the right child key is $AES_K(T2+2i)$. Using the mechanisms as set forth in appendix A.8 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 \parallel binary operator is used to denote concatenation. The notation $HMAC_SHA1\{k\}(s)$ is used to denote the computation of HMAC [RFC 2104] keyed by the key 'k' over the string 's' with SHA1 [NIST 180.2] 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 DEK 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. The $HMAC_SHA1$ is computed over a salt that is defined as follows. The DEK 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.

$\langle salt \rangle = BCI$

$DEK = HMAC_SHA1_128\{NK8 \parallel NK9 \parallel NK10 \parallel NK11 \parallel NK12 \parallel NK13 \parallel NK14\}(\langle salt \rangle)$.

Notes:

- 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 $HMAC_SHA1_128\{D1 \parallel D5\}(\langle salt \rangle)$. 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. Please refer to section A.8 for an overview of the function F_{ZMB} .

- 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 (e.g. 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 broadcast 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 BGK 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:

1. Devices owned by the same user are not assigned to the same group.
2. 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.

5.7 Interoperability with Alternative Implementations of the Functionality of Rights Management Layer and Registration Layer

All devices **SHALL** implement the rights management and registration layers as specified in chapters 6.3 and 6.4 of this specification. In addition, devices **MAY** contain alternative implementations of the functionality of these layers and Rights Issuers **MAY** make use of these alternative implementations, subject to the requirements below.

Alternative implementations of the functionality of the registration layer and rights management layer **SHALL** provide the following:

- A means to provide a collection of information called an RO from the RI to the device per service or programme that the device is entitled to consume. This collection of information consists of the following data:
 - The ID of the RI (riID).
 - The CID or BCI of the programme or service associated with the RO. If not apparent from the CID, the RO **SHALL** have an indication of whether the CID is associated with a programme or a service.
 - Optionally information regulating the consumption of the programme or service associated with the RO.
 - If the CID is for a programme, an RO associated with that programme **SHALL** contain key material for transferring the values of the PEK and PAK for that programme to the device. This key material **SHALL** be protected for confidentiality.
 - If the CID is for a service, an RO associated with that service **SHALL** contain key material for transferring the values of the SEK and SAK for that programme to the device. This key material **SHALL** be protected for confidentiality.
 - The combination of all of the above information **SHALL** be protected for integrity.

- A means to understand the permissions and constraints that are carried in the KSM, if any;
- A means to combine the permissions and constraints obtained from the KSM with the optional information in the RO for regulating consumption for making the decision to grant or deny a particular requested access to a particular programme or service;
- A means to provide from the RI to the device all information that is required for the key stream layer to decrypt the programme(s) or service(s) that the RI allows the device to access and providing this decryption information to the key stream layer for a particular programme or service if the combined permissions and constraints grant the requested form of access to this programme or service.

The device SHALL check all information it receives from the RI for integrity. In addition, it SHALL protect all cryptographic key information (e.g. all decryption and authentication keys) for confidentiality.

The specifications in this section define an interoperability point between the key stream layer and the rights management layer. This interoperability point not only allows for horizontal competition between providers of the Service Protection System according this specification including chapters 6.3 and 6.4, but also with providers of arbitrary, e.g. proprietary, solutions in a standards-compatible way.

6 The Four-Layer Model for Service and Content Protection

6.1 Traffic Layer

For protection of media streams on the traffic layer, IPsec or SRTP can be used, as defined in the following subsections.

6.1.1 IPsec

IPsec fulfils both the criterion to be bearer-agnostic and to be universally usable for all types of IP-based services. The broadcast network MAY use IPsec to protect broadcast services. All devices SHALL support IPsec.

The IPsec implementation in the device SHALL be such that it does not interfere with the usage of IPsec for applications other than data broadcasting. 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

The selectors can contain wildcards, ranges or point values, but all the other parameters SHALL be exactly defined. For transport mode, all address selectors SHALL be point values and the destination address selector SHALL match the destination IP address of the SA. An IPsec SA SHALL be uniquely identified by a destination IP address and SPI pair.

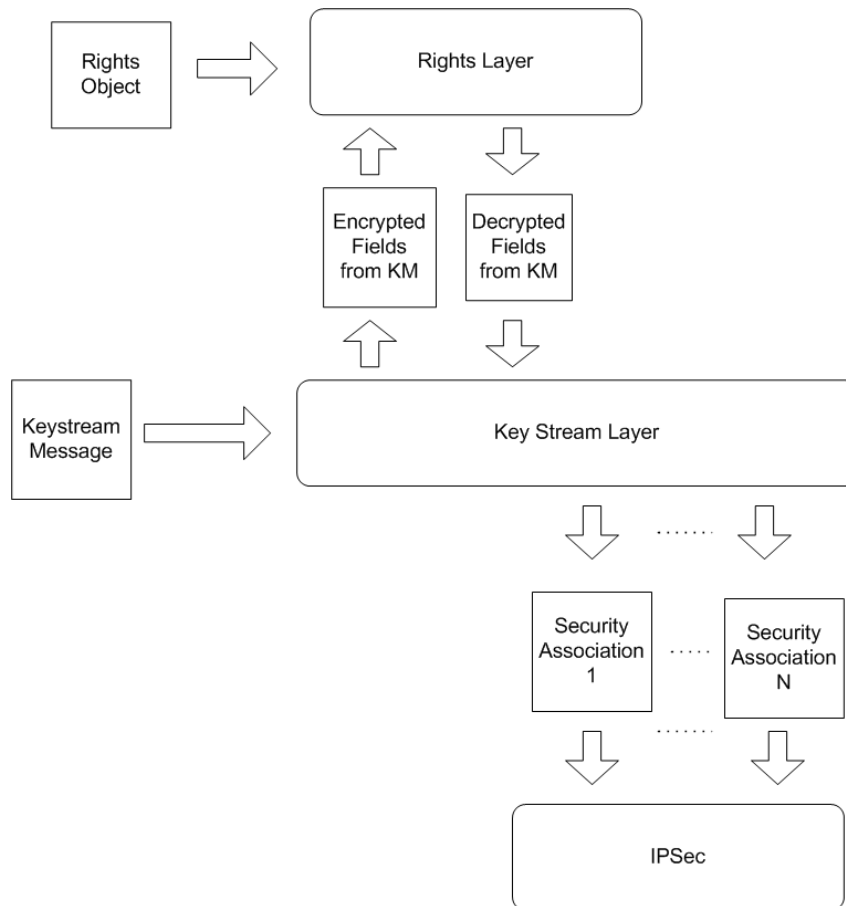


Figure 24: IPsec Security Association Elements

The figure above shows the different objects and elements involved in instantiating IPsec security associations. The instantiation of security associations is performed by the key stream layer and is driven by key stream messages and Rights Objects. Given a key stream message, 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 examines if that Rights Object would be able to decrypt the fields in the key stream message. If the rights layer does find a suitable rights object, then 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, then the key stream message **SHOULD** be silently dropped.

6.1.1.1 Selectors

Selectors are provided by the key stream protocol. The requirements for the selectors are that the IP addresses are point values and the destination IP address is equal to the destination address in the security association.

6.1.1.2 Encapsulation Protocol and mode

If IPsec is used for encryption of broadcast services, the protocol and mode **SHALL** be ESP in Transport Mode, according to [RFC 2401] and [RFC 2406]. Other IPsec encapsulation protocols or modes **SHALL NOT** be used.

6.1.1.3 Encryption Algorithm

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

6.1.1.4 Authentication Algorithm

The authentication algorithm for IPsec ESP SHALL be HMAC-SHA-1-96, as defined in [RFC 2104] and [RFC 2404]. Other authentication algorithms or truncations SHALL NOT be used.

A broadcast 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.

6.1.1.5 Security Association Management

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

- The traffic encryption key (TEK) contained in the key stream message SHALL be used as the key for the ESP encryption.
- The traffic authentication key (TAK) contained in the key stream message SHALL be used as the key for the ESP message integrity code if authentication is used.
- The IPsec implementation SHALL be able to manage security associations relating to the key stream messages separately from those managed manually or by any other protocol such as IKE. This implies the ability to identify whether an SA relates to key stream messages.
- The IPsec security policy (SP) SHALL be provided by the service guide, in the form of associations between encrypted IP flows and the key stream that carries the encryption keys. Security associations relating to key stream messages SHALL be prioritised lower than those security associations that have a locally defined policy or a policy that is provided by a trustworthy party. [Note to the editor: This sets a requirement on the ESG and must be checked once the ESG specification is complete and available.]
- Security associations relating to key stream messages are simplex and SHALL be applied only to inbound traffic on the recipient side.

The re-keying of existing security associations by the key stream layer 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 security associations for a particular set of selectors.
- The IPsec implementation SHOULD provide a least-recently-instantiated mechanism for cleaning up security associations as resources reserved for IPDC IPsec security associations are exhausted.
- The number of IP datacasting security associations required to exhaust the resources such that the cleanup mechanism is triggered SHOULD be 3 per service key per set of IP selectors.
- A device SHOULD be able to rekey any security association at least for every 20 received ESP packets without a significant loss in performance. This rekey consists of installing a new security association with a defined set of selectors, and possibly, eliminating an old security association with an equal set of selectors. Both security associations in this case are managed by the Key Stream Layer.

Note however, that for a broadcaster it is not recommended to rekey existing security associations for every 20 packets, as the amount of traffic one can place in 20 packets varies heavily with the maximum packet size. Also the impact on the device in terms of time is hard to estimate, as the timing between packets may be significantly altered in a broadcasting environment. Therefore a broadcaster SHALL NOT rekey an IPsec SA more often than every two seconds at the point of sending the key stream messages.

6.1.2 SRTP

The broadcast network MAY use SRTP to protect broadcast services. Devices SHALL support SRTP.

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
- MAC method definition
- list of received packets
- MKI indicator bit
- length of the MKI field
- value of currently active MKI
- array of secret master keys (MK)
- array of counter 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 SSRCs, destination network address and destination transport port number.

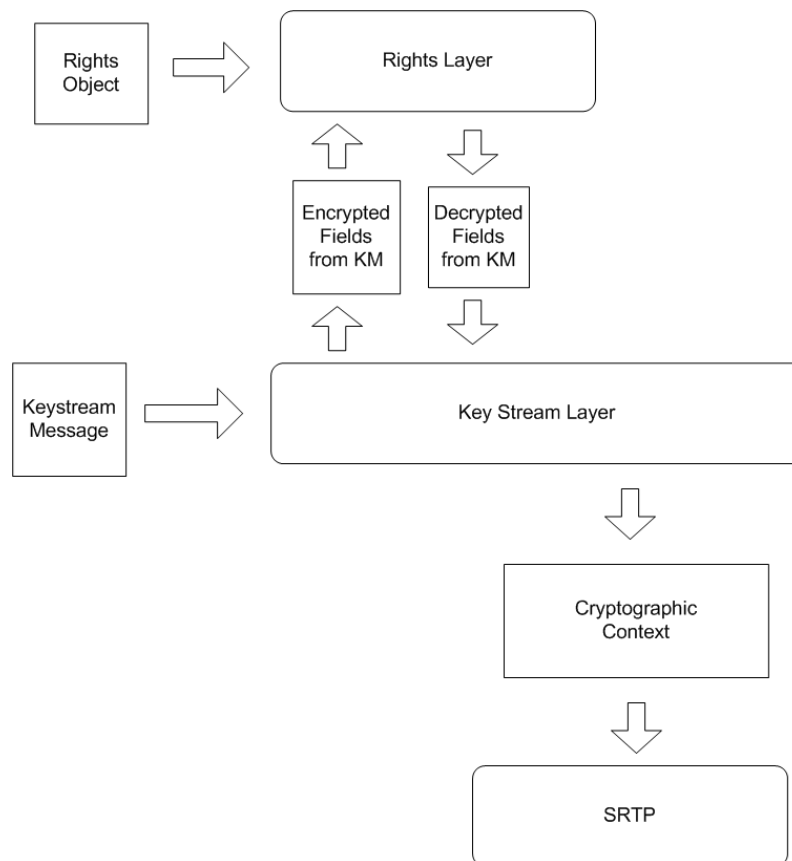


Figure 25: SRTP Cryptographic Context Management

6.1.2.1 Key Management

The IPDC 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 traffic key material contained in the key stream message SHALL be used as the SRTP master key.

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

The key stream SHALL provide and update the cryptographic contexts to the SRTP implementation. Note that some fields are initialised and/or managed internally, 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 for which a MAC is computed over (assuming authentication is used). The ROC values between the sender and the device SHALL be synchronized. The 48-bit packet index value is included in the (implicit) IV for the AES-CM encryption, and therefore the ROC is needed to encrypt/decrypt a packet.

Because the SRTP key-derivation rate is not used and the <From,To> values are also not used, the SRTP crypto context will be rekeyed by the key stream layer. 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 key stream messages.

6.1.2.2 Encryption Algorithm

The encryption algorithm for SRTP packets SHALL be AES-128-CTR, as defined in [RFC 3711]. Other encryption algorithms or key sizes or chaining modes SHALL NOT be used.

6.1.2.3 Authentication Algorithm

The authentication algorithm for SRTP SHALL be HMAC-SHA-1-80, as defined in [RFC 2104] and [RFC 3711]. Other authentication algorithms or truncations SHALL NOT be used.

The broadcast system MAY use authentication. If no authentication is desired, the NULL authentication algorithm SHALL be specified. In this case, also replay protection SHALL NOT be performed by the device.

6.2 Key Stream Layer

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

6.2.1 Key Stream Message (KSM)

Each KSM SHALL be encapsulated in exactly 1 UDP packet.

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.

Table 3: Format of Key Stream Message

Key_Stream_Message_Description	Length	Type
key_stream_message() {		
selectors_and_flags {		
protocol_version	4	uimsbf
reserved_for_future_use	4	bslbf
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 (traffic_protection_protocol == KSM_ALGO_SRTTP) {		
master_key_index_length	8	uimsbf
master_key_index	8*length	uimsbf
number_of_media_flows	8	uimsbf
for (i = 0; i < number_of_media_flows; i++) {		
synchronization_source	32	uimsbf
rollover_counter	32	uimsbf
}		
}		
encrypted_traffic_key_material_length	8	uimsbf
encrypted_traffic_key_material	8*length	bslbf
if (next_traffic_key_flag == KSM_FLAG_TRUE) {		
next_encrypted_traffic_key_material	8*length	bslbf
}		
reserved_for_future_use	5	bslbf
traffic_key_lifetime	3	uimsbf
if (timestamp_flag == KSM_FLAG_TRUE) {		
timestamp	40	mjdutc
}		
if (programme_flag == KSM_FLAG_TRUE) {		
programme_selectors_and_flags {		
reserved_for_future_use	6	bslbf
access_criteria_flag	1	uimsbf
permissions_flag	1	uimsbf
}		
if (access_criteria_flag == KSM_FLAG_TRUE) {		
reserved_for_future_use	4	bslbf
parental_rating	4	uimsbf
number_of_access_criteria_descriptors	8	uimsbf
access_criteria_descriptor_loop() {		
access_criteria_descriptor()		
}		
}		
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
}		
}		

6.2.1.1 Descriptors for access_criteria_descriptor_loop

Table 4: Descriptors for access_criteria_descriptor_loop

Access_Criteria_Descriptor	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 specification. The device SHALL ignore access criteria descriptors that it doesn't support.

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

6.2.1.2 Constants

Table 5: Constants in Key Stream Message

Name	Value
KSM_ALGO_IPSEC	0
KSM_ALGO_SRTP	1
KSM_FLAG_FALSE	0
KSM_FLAG_TRUE	1

6.2.1.3 Coding and Semantics of Attributes

protocol_version – indicates the protocol version of this key stream message.

The device SHALL ignore messages that have a protocol_version number it doesn't support. The value of the protocol_version of this message is set to 0x0 (i.e. the original format).

Note: If set to 0x0 the format specified in the this version of the specification is used. If set to anything else than 0x0, then the format is beyond the scope of this version of the specification.

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)

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 second 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 minute 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 minute.

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 is associated with the current TEK. If the next traffic key flag is set to 1, the SPI associated with the “next TEK” is implicitly defined as SPI+1.

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 [RFC3711] 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.

number_of_media_flows – specifies how many RTP media flows are protected by the key stream:

For each of the media flows, the SRTP roll-over counter needs to be signalled.

synchronization_source – identifies an RTP media flow to which the associated roll-over counter applies.

rollover_counter – signals the current roll-over counter of the RTP media flow identified by the synchronization source.

The roll-over counter is an extension of the sequence number contained in the SRTP packet. It can be different for each SRTP-protected media flow, even if the same key stream is used. Therefore, to allow devices instant service access, the current value of the roll-over counter for each media flow is signalled in the KSM.

Whenever the sequence number of one of the media flows rolls over, a new crypto period SHALL be started, with an incremented MKI, and the new ROC for the media flow in question. The network SHALL ensure that such a ROC-triggered change of the crypto period doesn't violate the lower bound of crypto period durations.

A device that is already tuned to a particular channel SHALL locally keep track of the ROC values and increment them when the RTP sequence number wraps around (this is really an SRTP requirement).

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:

- 1) 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 described in A.9.
- 2) SRTP:
The master key is identical to the decrypted traffic key material. If no traffic authentication is used, the master key has a length of 16 bytes; if traffic authentication is used, 36 bytes. How the TEK and TAK are derived from the master key is defined by SRTP.

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 similar to `encrypted_traffic_key_material` attribute.

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

If `<traffic_key_lifetime>` is *n*, then the actual lifetime is **2ⁿ** seconds, as presented in the following table:

Table 6: 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 optimisations.

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 a 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 Annex A.4. This 40-bit field contains the timestamp of the key stream message in Universal Time, Co-ordinated (UTC) and Modified Julian Date (MJD). This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit Binary Coded Decimal (BCD).

EXAMPLE 1: 93/10/13 12:45:00 is coded as "0xC079124500".

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.

Access criteria cannot change during a programme, i.e. as long a programme key is valid.

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

parental_rating – is the parental rating of the programme, for which the following formula SHALL hold:

$$\langle \text{parental_rating} \rangle = \langle \text{age_limit} \rangle - 3$$

The content transported within the programme can only be consumed by someone aged <age_limit> years or older.

Note: this is the definition provided by ETSI EN 300 468 for the parental_rating_descriptor in DVB systems.

number_of_access_criteria_descriptors – indicates the number of access criteria descriptors.

permissions_category – indicates the permissions category for the programme:

0x00 - no permissions category, service RO applies as such,
 0x01...0x3F - permissions_category is included in the post-acquisition permissions lookup, and
 0x40...0xFF - reserved for future standardization.

If permissions_category is in the range 0x01...0x3F,

- in case of ICRO, the device SHALL use as service_CID for post-acquisition permissions lookup the text string

```
service_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 case of BCRO, the device SHALL look up the permissions specified in the service BCRO for the asset that has a matching permissions_category field.

If permissions_category is in the (reserved for future standardization) range 0x40...0xFF, and device does not support it, device SHALL drop (i.e. ignore) all post-acquisition permissions (like play, redistribute etc.) indicated in the service RO, or if device cannot do such permissions 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.

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 which 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 = socID + "#P" + serviceBaseCID + "@" + hex(programme_CID_extension)
programme_BCI = hash(socID + "#P" + serviceBaseCID + "@") + programme_CID_extension
```

The socID and serviceBaseCID are string values and are expected to be part of the service guide. 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. [Note to the editor: This sets a requirement on the ESG and must be checked once the ESG specification is complete and available.]

The hex() function is a hexadecimal presentation of the parameter containing hexadecimal characters 0-9 and a-f (in lowercase) with possible preceding zeros. EXAMPLE: for a 16 bit value 2748, hex() returns "0abc". Note that two characters are always generated for each byte.

The hash function for the construction of programme_BCI is defined in A.5, where BCI is defined. It doesn't depend on the contents of the KSM, and can thus be pre-computed.

programme_MAC – is the HMAC-SHA-1-96 according to RFC 2104 and 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 directly decrypt the traffic key material.

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 which allows to identify 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 = socID + "#S" + serviceBaseCID + "@" + hex(service_CID_extension)

service_BCI = hash(socID + "#S" + serviceBaseCID + "@") + service_CID_extension
```

The socID and serviceBaseCID are string values and are expected to be part of the service guide. 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. [Note to the editor: This sets a requirement on the ESG and must be checked once the ESG specification is complete and available.]

The hex() function is a hexadecimal presentation of the parameter containing hexadecimal characters 0-9 and a-f (in lowercase) with possible preceding zeros. EXAMPLE: for a 16 bit value 2748, hex() returns "0abc". Note that two characters are always generated for each byte.

The hash function for the construction of service_BCI is defined in A.5, where BCI is defined. It doesn't depend on the contents of the KSM, and can thus be pre-computed.

If the permissions_category field is present and has a nonzero value, the Service_CID of the service is constructed as specified above (at description of the permissions_category field).

service_MAC – is the HMAC-SHA-1-96 according to RFC 2104 and 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 doesn't 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.

6.2.2 Key Stream Discovery

The access description to a particular service which is distributed as part of the Service Guide is assumed to contain a media description for each IP flow of the media service itself.

Based on the basic assumption that the service can't be consumed (because the used IP addresses, codecs, and other "technical" parameters are not known) unless the access description is present in the device; the access description will also carry the static security-related parameters of the service or of a session of the 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.

The following access information pertaining to the traffic key stream SHALL be added to the access description of the service:

port_of_key_stream – is the port number of the UDP stream carrying the KSM flow.

IP_address_of_key_stream – is the IP address on which the key stream is transported, which can be an IPv4 or and IPv6 address.

To the service entity itself, the following information SHALL be added:

serviceBaseCID – is the static part of all CIDs of SEAKs and PEAKs pertaining to the service.

6.3 Rights Management Layer

All devices SHALL implement the rights management layer as specified in this chapter. Devices MAY additionally implement alternative schemes for the functionality of the rights management and registration layers - see chapter 5.7 for details.

In case of **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 broadcast 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 case of **pay-per-view**, the PEAK associated with a pay-per-view event is delivered to the authorized device directly within an RO that is bound to a device, a unique group, a broadcast 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 the interactivity channel (i.e. the ID of an ICRO) linked with subscription (Service RO) or pay-per-view (Programme RO), and bound to a device or to a domain, and SHALL be constructed respectively as follows:

```
deviceRoID = "E" + deviceID + "/" + socID + "#I" + purchaseItemID + "@" +
             hex(rekeying_period_number)
```

```
domainRoID = "O" + domainID + "/" + socID + "#I" + purchaseItemID + "@" +
             hex(rekeying_period_number)
```

The `hex()` function is a hexadecimal presentation of the parameter containing hexadecimal characters 0-9 and a-f (in lowercase) with possible preceding zeros. EXAMPLE: for a 16 bit value 2748, `hex()` returns "0abc". Note that two characters are always 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 section 6.3.4.2).

6.3.1 Requirements for Service ROs

A Service RO SHALL contain at least one asset, i.e. 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 6.2.1.3.

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)

The SAK (160 bits, since HMAC-SHA-1-96 is used to calculate the service_MAC) is derived from the SAS, as described in A.9.

The Service RO SHALL contain a rights expression defining a “datetime” 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. The rights expression in the Service RO SHALL NOT contain any further constraints to the “access” permission.

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 governing post-acquisition usage, and the expressions or the permissions SHALL be enforced.

The post-acquisition permissions MAY be different for different programmes, depending on the permissions_category value signalled in KSM (see chapter 6.2). 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.

If the RO does not contain a rights expression, the content distributed within the programme can freely be used by the device after authorized reception.

[Note to the editor: The “access” permission is a new permission that is expected to be added 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. Should such an extension not be possible, the “execute” permission could be used instead.]

6.3.2 Requirements for Programme ROs

A Programme RO SHALL contain exactly one asset, i.e. a (<programme_CID/BCI>, <wrapped_PEAK>) pair. The CID/BCI pertaining to the programme SHALL be constructed as specified in 6.2.1.3.

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)

The PAK (160 bits, since HMAC-SHA-1-96 is used to calculate the programme_MAC) is derived from the PAS, as described in A.9.

The Programme RO SHALL contain a rights expression defining a “datetime” 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. The rights expression in the Programme RO SHALL NOT contain any further constraints to the “access” permission.

If and only if post-delivery content protection is used in a system, the Service RO MAY contain further rights expression governing post-acquisition usage, and the expression or the permissions SHALL be enforced.

If the RO does not contain a rights expression, the content distributed within the programme can freely be used by the device after authorized reception.

[Note to the editor: The “access” permission is a new permission that is expected to be added 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. Should such an extension not be possible, the “execute” permission could be used instead.]

6.3.3 Delivery of ICROs over Interactivity Channel

When the Interactivity Channel is used to deliver Rights Objects, all aspects of Rights Object delivery are governed by OMA DRM 2.0 specifications [OMA-DRM-DRM]. Rights Objects delivered over the interactivity channel are referred to as Interactivity Channel Rights Objects (ICRO).

6.3.4 Delivery of BCROs over Broadcast Channel

6.3.4.1 Broadcast of BCRO Objects

BCRO Objects SHALL be broadcast within an RI Service, as defined in chapter 7. The RI Service streams used by individual Rights Issuers are identified by the ESG.

6.3.4.2 Format of a Broadcast Rights Object (BCRO)

The following tables define the format of a BCRO. The *asset*, *permission* and *constraint* object correspond in their meaning to their counterparts in [OMA-DRM-DRM]. The *action* object corresponds to the allowed elements in the [OMA-DRM-REL] 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 7: 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
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 A.14.

version: 3-bit flag which indicates the version of the BCRO message format. If 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 4096 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 timestamped.

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 8 lists all four possible address modes.

Table 8: address_mode

address_mode	description
0x0	addressing whole of unique group
0x1	addressing of broadcast 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 bit of the address field. If the BCRO is addressed to a unique device in a group then the LSB of the address field is the MSB 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 BCRO addresses a subset of a unique group (address_mode 0x1) then 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 broadcast group. If group_size_flag is 0 then the position in the group is directly given by the position_in_group field. If group_size_flag is 1 then 9 bit 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 MSB. 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.

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 X509PKISHash in [OMA-DRM-DRM].

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 Annex A.4. This 40-bit field contains the timestamp of the BCRO in Universal Time, Co-ordinated (UTC) and Modified Julian Date (MJD). This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit Binary Coded Decimal (BCD).

refresh_time: The refresh_time specifies the time when the device should acquire a new BCRO. It does not specifies 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 Annex A.4. This 40-bit field contains the expiry time of the BCRO in Universal Time, Co-ordinated (UTC) and Modified Julian Date (MJD). This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit Binary Coded Decimal (BCD).

EXAMPLE 1: 93/10/13 12:45:00 is coded as "0xC079124500".

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 which 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.

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

MAC: This is the authentication code calculated over all bytes before this field in the BCRO using HMAC-SHA-1-96 (see [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 Annex A.9.3.

6.3.4.2.1 Format of the asset object

Table 9: asset format

Field	length	type
asset() {		
BCI	96	bslbf
key_flag	1	
key_type	1	uimsbf
reserved_for_future_use	2	uimsbf
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	uimsbf
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 Annex A.5. 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 program encryption key (PEK)

inherit_flag: 1-bit flag indicating whether inheritance is used. If 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 10. 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 10: asset_type

asset_type	Description
0x0	broadcast stream protected IPsec or SRTP as defined in this specification
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 uniquely identify the parent rights object.

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 nonzero, 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). Note that 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 its 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 then this field contains the encrypted SEAK, the service encryption key (SEK) concatenated with the Service Authentication Seed (SAS). 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.

Table 11: Mapping of address_mode to keys

address_mode	Keys used
0x0 (unique group)	UGK (Unique Group Key)
0x1 (broadcast group)	Deduced SEK decryption key (based on bit_access_mask and zero message broadcast group keys)
0x2 or 0x3 (unique device)	UDK (Unique Device Key)
0x4 (OMA domain)	LDK (Local Domain Key)

encrypted_program_encryption_authentication_key: If key_type is set to 1 then this field contains the encrypted PEAK, the program encryption key (PEK) concatenated with the program authentication seed (PAK). 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.

Table 12: Mapping of address_mode to keys

address_mode	Key(s) used to decrypt field
0x0 (unique group)	UGK (Unique Group Key)
0x1 (broadcast group)	Deduced PEK decryption key (based on bit_access_mask and zero message broadcast group keys)
0x2 or 0x3 (unique device)	UDK (Unique Device Key)
0x4 (OMA domain)	LDK (Local 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.

Table 13: Mapping of address_mode to keys

address_mode	Key(s) used to decrypt field
0x0 (unique group)	UGK (Unique Group Key)
0x1 (broadcast group)	Deduced CEK decryption key (based on bit_access_mask and zero message broadcast group keys)
0x2 or 0x3 (unique device)	UDK (Unique Device Key)
0x4 (OMA domain)	LDK (Local Domain Key)

6.3.4.2.2 Format of the permission object

Table 14: permission format

Field	length	type
<code>permission() {</code>		
<code>number_of_assets</code>	6	uimsbf
<code>constraint_flag</code>	1	uimsbf
<code>actions_flag</code>	1	uimsbf
<code>for(i = 0; i < number_of_assets; i++){</code>		
<code>asset_index</code>	8	uimsbf
<code>}</code>		
<code>if(constraint_flag == 1){</code>		
<code>constraint()</code>		
<code>}</code>		
<code>if(actions_flag == 1){</code>		
<code>number_of_actions</code>	8	uimsbf
<code>for(i=0; i < number_of_actions; i++){</code>		
<code>action()</code>		
<code>}</code>		
<code>}</code>		
<code>}</code>		

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.

constraint_flag: 1-bit flag which 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.

action_flag: 1-bit flag. When set to 1, 1 or more actions are contained in 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).

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

6.3.4.2.3 Format of the action object

Table 15: 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 16.

Table 16: action_type

action_type	description
0x00	PLAY_ACTION
0x01	DISPLAY_ACTION
0x02	EXECUTE_ACTION
0x03	PRINT_ACTION
0x04	EXPORT_ACTION
0x05	ACCESS_ACTION
0x06-0x7F	reserved for future use

constraint_flag: 1-bit flag which 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.

6.3.4.2.3.1 Action definitions

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-DRM-REL].

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

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

A system_constraint object contained in a ACCESS_ACTION object is used to specify 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 MUST NOT grant access according to an ACCESS_ACTION to content that cannot be rendered in this way.

Note that the device MUST 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 MUST be utilized instead.

6.3.4.2.4 Format of the constraint object

Table 17: constraint format

Field	length	type
constraint() {		
number_of_constraints	4	uimsbf
constraint_descriptor_length	12	uimsbf
for(i=0;i<number_of_constraint;i++){		
constraint_descriptor()		
}		
}		

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

constraints_descriptor_length: length of all constraint descriptors in bytes which follow this field.

Table 18: constraint_descriptor format

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 19.

Table 19: 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
0x07-0xFF	reserved for future use

6.3.4.2.4.1 Count constraint descriptor

Table 20: 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 [OMA-DRM-REL].

6.3.4.2.4.2 Timed count constraint descriptor

Table 21: 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 [OMA-DRM-REL]

6.3.4.2.4.3 Date-time constraint descriptor

Table 22: 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_date	40	mjdutc
}		
if(end_flag == 1){		
end_date	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.

start_time: Time field with the semantics of ‘not before’ time for a permission. The start_time must be before the end_time if present. See [OMA-DRM-REL].

end_time: Time field with the semantics of ‘not after’ time for a permission. The end_time must be after the start_time if present. See [OMA-DRM-REL].

6.3.4.2.4.4 Interval constraint descriptor

Table 23: 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 bit. See [OMA-DRM-REL] for a more detailed description

6.3.4.2.4.5 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.

Table 24: 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 bit. See [OMA-DRM-REL] for a more detailed description

6.3.4.2.4.6 Individual constraint descriptor

Constraint 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.

Table 25: 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.

id_type: Tag identifying format of the individual_id as listed in Table 26.

Table 26: 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 [OMA-DRM-REL] for a more detailed description

6.3.4.2.4.7 System constraint descriptor

Constraint used identify systems to which the content and rights objects are allowed to be exported to.

Table 27: system_constraint_descriptor

Field	length	type
system_constraint_descriptor() {		
constraint_tag	8	uimsbf
length	8	uimsbf
system_id	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 of the system name as registered with OMNA. See [OMA-DRM-REL] for a more detailed description.

6.3.4.2.4.8 Metering constraint descriptor

The metering_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 metering_constraint_descriptor indicate how “much” consumption of DRM content requires how many tokens need to be consumed from the token store.

Table 29: metering_constraint_descriptor

Field	length	type
metering_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
}		

length: The number of bytes following this field.

token_constraint_type: If the value of this field equals 0x0 (COUNT), the consumption of the DRM content must 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), 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 must 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.

6.4 Registration Layer

All devices SHALL implement the registration layer as specified in this chapter. Devices MAY additionally implement alternative schemes for the functionality of the rights management and registration layers - see chapter 5.7 for details.

6.4.1 RI Context

There are two types of RI context, being:

- RI context for interactive mode of operation. This is specified in [OMA-DRM-DRM], whereas some details are listed in section 6.4.2.
- RI context for broadcast mode of operation. This is specified in this specification in section 6.4.3

Please note that both types of RI context are different from each other.

6.4.2 Interactive mode of operation

Registration is according to [OMA-DRM-DRM] Chapter 5.4.2.

6.4.3 Broadcast mode of operation

6.4.3.1 Protocol overview

The theory of operation (refer to section 5.3) results in the specification of several protocols:

- offline protocols (from device to RI)

protocol	section	Purpose
offline Notification of Detailed Devicedata protocol	6.4.3.2	Registration
offline Notification of Short Devicedata protocol	6.4.3.3	inform RI by action request codes

- 1-pass protocols (from RI to device)

protocol	section	Purpose
1-pass binary Push Device Registration protocol	6.4.3.4	transmit registration data to device
1-pass binary Inform Registered Device protocol	6.4.3.5	inform device via messages

- supporting protocols for registration

protocol	section	Purpose
Unique Device Number (UDN) protocol	6.4.3.6	make registration data robust (part of offline notification of detailed device data)

6.4.3.2 offline Notification of Detailed Device data protocol

Note: This protocol is also known as the “offline NDD protocol”, short for offline Notification of Detailed Data protocol.

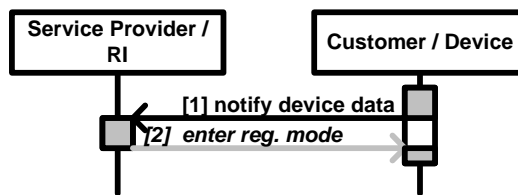


Figure 26: offline NDD protocol

N.B: notification of device data is performed off-line. The device data (device_data_inform() message) is defined in section 6.4.3.7.

Explanation of the protocol:

- 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 (refer to section 6.4.3.7.1 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, or else.

An example of a displayed message follows, where the following information is reported back to the RI¹:

¹ 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 specification. 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 (please note: the short UDN will only be displayed after the first registration, when that data **MAY** available for display).

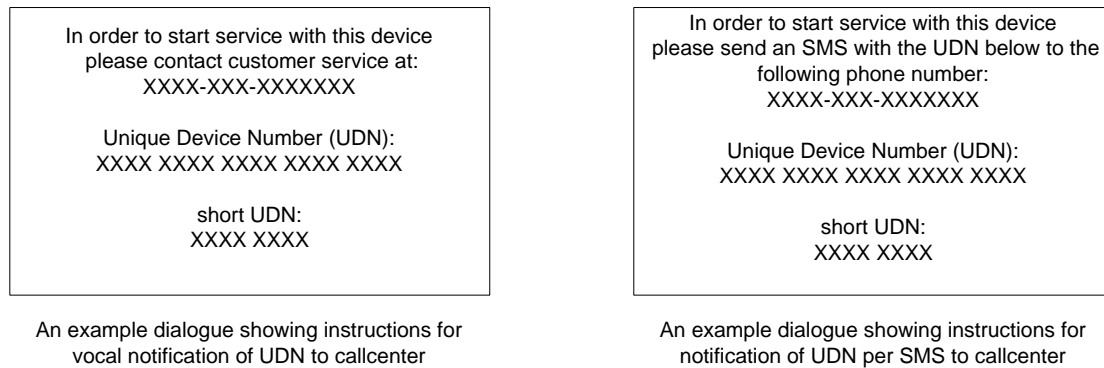


Figure 27: samples of notification displays

- 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 offline Notification of Detailed Devicedata protocol. The device data to notify SHALL be reduced by a special protocol (refer to section 6.4.3.6).
- After the notification of the device data, user needs to put the device into registration mode [2]. When put into registration mode, device SHALL start to listen for the device registration data for a limited time.

6.4.3.3 offline Notification of Short Devicedata protocol

Note: This protocol is also known as the “offline NSD protocol”, short for offline Notification of Short Data protocol.

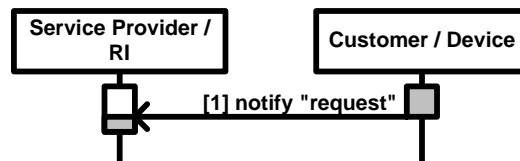


Figure 28: offline NSD protocol

Note: Notification of device data is performed off-line. Refer to Table 29 for an overview of the possible “requests”.

Explanation of the protocol:

- The user may notify a short decimal code called the action request code (ARC) to the RI via offline methods (e.g. telephone call or SMS or else). The code SHALL be constructed as follows:

Short_udn	Action_code	Checksum
-----------	-------------	----------

Figure 29: Action Request Code (ARC)

Note that for some of the ARCs (e.g. the ARC token_consumption_report), the user MAY have to notify more digits to the RI than the ones of the ARC.

Table 28: NSD action request code fields

ARC fields	Length (digits)	supporting up to
short_udn	8	100 Million devices
action_code	2	99 action codes
checksum	2	

This totals to 12 digits. The fields are explained below:

short_udn. The offline 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 `short_form_udn`) that is carried in the `device_registration_response()` message. The `short_form_udn` number is used to speed up the offline interaction with the RI. If this number is stored into the device, subsequent “requests” by the user of the device can be notified offline much quicker by using the `short_form_udn` number concatenated by a standardised action code.

Please note: In cases where the device needs to be identified uniquely in another network than it’s home network where it was registered, the `short_udn` cannot be used because the (new / different) RI does not have the `short_udn` in it’s database. In this case the only possibility for the hosting RI to identify the device uniquely would be via the `long_udn`. It is the responsibility of the device to decide when it is appropriate to use the `long_udn` instead, for example by comparing the Service Operations Centre (SOC) ID received with the SOC ID remembered from registration.

action_code. Following the `short_udn` the user of the device can notify an action code to the RI. The NSD protocol defined in this specification SHALL use following `action_codes` to construct the ARC:

Table 29: NSD action types

action type	action code (d)	described in section
re-registration (only at same RI)	{0d01}	6.4.3.3.1
resend BCRO	{0d02}	7.9
reserved for future use	{0d03,...,0d09}	
join domain	{0d10,...,0d19}	6.4.3.3.
leave domain	{0d20,...,0d29}	6.4.3.3.
purchase	{0d30}, whereas content identification is supplied by ESG.	
token_consumption_report	{0d31,...,0d39}	6.4.3.3.
metering	{0d40,...,0d49}	
token_request	{0d50,...,0d59}	
notify DRM time drift	{{0d7}+{0d0,...,0d9},...,{0d8}+{0d0,...,0d9}}	6.4.3.3.
reserved for future use	{0d90,...,0d99}	

checksum. The constructed `short_udn` and `action_code` is appended by checksum digits. Please refer to section A.10 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 follows, where the following information is reported back to the RI²:

<p>In order to start the requested action please contact customer service at: XXXX-XXX-XXXXXXX</p> <p>action request code: XXXX XXXX XXXX</p>	<p>In order to start the requested action please send an SMS with the short request code (NSD) below to the following phone number: XXXX-XXX-XXXXXXX</p> <p>action request code: XXXX XXXX XXXX</p>
---	---

An example dialogue showing instructions for
vocal notification of ARC to callcenter

An example dialogue showing instructions for
notification of ARC per SMS to callcenter

Figure 30: samples of notification displays showing an ARC message

² 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 specification. 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 (please note: the short UDN will only be displayed after the first registration, when that data MAY available for display

6.4.3.3.1 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. (refer to section 6.4.3.4).

6.4.3.3.2 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.

6.4.3.3.3 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 join 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.

6.4.3.3.4 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 e.g. 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 follows, where the following information is reported back to the RI³:

In order to start the requested action
please contact customer service at:
XXXX-XXX-XXXXXXX

action request code:
XXXX XXXX XXXX

token consumption data:
XXXX XXXX XXXX XXXX XXXX

Figure 31: sample of token consumption reporting notification display

³ 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 specification. 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 MUST be included as defined above (please note: the short UDN will only be displayed after the first registration, when that data MAY available for display

6.4.3.3.4.1 Token consumption data definition

The token consumption data are defined below.

Table 30: 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 to 20digits. The fields are explained below:

tokens_consumed – This field contains the amount of tokens the device wished to report as consumed to the RI. See section 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 section 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 3 digits allow 1 out of 10^3 possible errors to remain undetected. The checksum algorithm used is the UDN checksum, see section A.10.1.

6.4.3.3.5 Notify DRM time drift

Time drift is expressed in minutes and rounded up to next multiple of 5 minutes. The range is 0..100 minutes, whereas value 69 will decode as timedrift ≥ 100 . Some examples of valid ARC values are given below:

E.g.1: Device notifies 4 minutes timedrift from newly received DRM time message: action code is 70.

E.g.2: Device notifies 38 minutes timedrift from newly received DRM time message: action code is 78.

E.g.3: Device notifies 235 minutes timedrift from newly received DRM time message: action code is 89.

6.4.3.4 1-pass binary Push Device Registration Protocol

Note: This protocol is also known as the “1-pass PDR protocol”, short for Push Device Registration protocol.

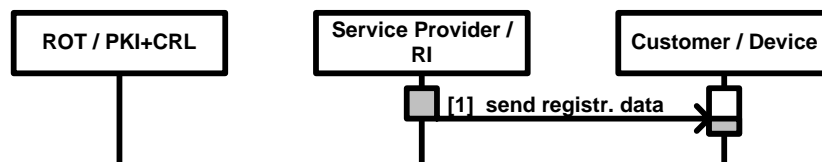


Figure 32: 1-pass PDR protocol – (first) device registration

Note: Transmission of registration data is performed on-line via the broadcast channel. The registration data (device_registration_response() message) is specified in section 6.4.3.7.2

Explanation of the protocol:

- 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 (refer to 6.4.3.7.2) or the domain_registration_response() message (refer to 6.4.4.4.1). The RI SHALL use the mechanisms described in section 7.6 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 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. 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.

6.4.3.5 1-pass binary Inform Registered Device Protocol

Note: This protocol is also known as the “1-pass IRD protocol”, short for Inform Registered Device protocol.

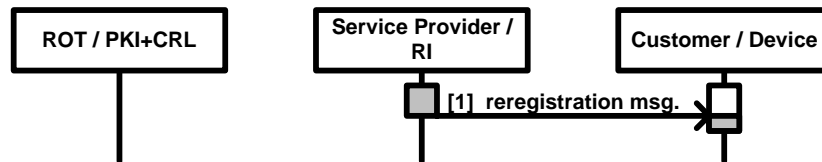


Figure 33: 1-pass IRD protocol – RI initiated message to device (here re-registration).

Explanation of the protocol:

- 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 31: Messages of the 1-pass IRD protocol

message name	for msg syntax refer to section	remark
force to re-register	6.4.3.7.3	
update RI certificate	6.4.3.7.4	in BCRO carousel
update DRM Time	6.4.3.7.5	in BCRO carousel
update contact number	6.4.3.7.6	in BCRO carousel
update domain	6.4.4.4.2	
force to join domain	6.4.4.4.3	
force to leave domain	6.4.4.4.6	
token delivery	6.4.5.4.1	

Note: The processing of each message will be discussed in following sections.

6.4.3.5.1 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 described in section 7.6 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 described below:
 - Accessing an ESG 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 offline 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 offline NDD protocol SHALL be executed.

6.4.3.5.2 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 described in section 7.6 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.

6.4.3.5.3 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 timeoffset SHALL be valid.
- The RI SHALL use the mechanisms described in section 7.6 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.

6.4.3.5.4 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 described in section 7.6 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).

6.4.3.5.5 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 described in section 7.6 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.

6.4.3.5.6 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 described in section 7.6 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.

6.4.3.5.7 update a domain

The RI can use this message to inform the device that he left a particular domain.

- The message SHALL contain a valid domain id.
- The RI SHALL use the mechanisms described in section 7.6 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.
- If the message is correct, the device SHALL delete the associated domain context.

6.4.3.6 Unique Device Number (UDN) protocol

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.

Following algorithm SHALL be used to construct a Unique Device Number (a.k.a. UDN):

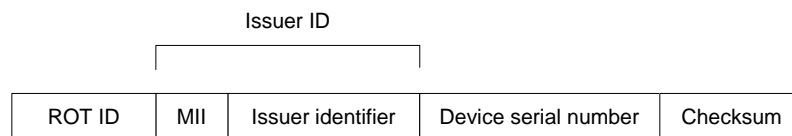


Figure 34: Unique Device Number

Table 32: UDN explanation

Field	Length (digits)	supporting up to
rot_id	3	1000 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 to 20 digits. The fields are explained below:

Every of 1000 ROT can issue 100.000 issuer ranges, from which every unique issuer can have 1 Billion devices issued.

rot_id - The first 3 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 following issuer categories:

Table 33: 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	

For 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 3-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 1Billion).

checksum - The final digits of the device ID number are check digits, akin to a checksum. The 3 digits allow 1 out of 10^3 possible errors to remain undetected. Please refer to section A.10 for an explanation of the algorithm.

6.4.3.6.1 Message syntax

The 20 digits of the UDN are encoded in BCD format into the longform_udn(). The message syntax is specified below:

Table 34: longform_udn

fields	length	type
longform_udn(){		
rot_id	12	bslbf
mii	4	bslbf
issuer_identifier	16	bslbf
device_serial_number	36	bslbf
checksum	12	bslbf
}		

6.4.3.7 Binary messages

6.4.3.7.1 Device data – device_data_inform() message

6.4.3.7.1.1 Message description

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.

Table 35: Notify device data message parameters

Device_Data_Inform()		
parameter	(M)andatory / (O)ptional ⁴	Remark
version	M	
contact_nr	O	
longform_udn()	M	

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_nr - 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 (refer to section 6.4.3.2 for an example). This number could also be provided in other ways, like a leaflet in the package of the device, a commercial channel which is viewed after selection of a free to air channel via the ESG 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. Refer to section 6.4.3.6 for details.

6.4.3.7.1.2 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 follows:

Table 36: 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

6.4.3.7.2 Registration data – device_registration_response() message

6.4.3.7.2.1 Message description

Using the 1-pass PDR protocol the RI SHALL send a device_registration_response() message with the registration data to the device as specified below:

⁴ key: (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.

Table 37: message description

Device Registration Response()		
Parameter name	(M)andatory / (O)ptional ⁵	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
broadcast_group_key_flag	M	device specific, not encrypted
signature_type_flag	M	global, not encrypted
short_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
broadcast_group_key	O	device specific, encrypted
unique_device_key	O	device specific, encrypted
unique_device_filter	M	device specific, encrypted
ri_authentication_key	M	device specific, encrypted
local_domain_key	O	device specific, encrypted
shortform_domain_id	M	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

message_tag - This parameter identifies the type of the message. Refer to section A.14 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 doesn't support. The value of the protocol_version of this message is set to 0x0 (i.e. the original format).

If set to 0x0 the format specified in the this version of the specification is used. If set to anything else than 0x0, then the format is beyond the scope of this version of the specification

longform_udn() - The long form of the UDN. Refer to section 6.4.3.6 for details.

status - The status parameter SHALL indicate one of the values explained in the following table. The device SHALL ignore messages with other error values.

⁵ key: (O)ptional means that the user of the message MAY include the parameter in the message, but the device MUST support the interpretation of the parameter. (M)andatory means that the user of the message SHALL include the parameter in the message.

Table 38: 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 (offline 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 (refer to section 6.4.3.5.1)

certificate_version - is a numerical representation of the version of the RI certificate. Using the *certificate_version* parameter the device can decide if it is needed to update the RI certificate (if it was stored before).

Table 39: description of certificate_version parameter

Parameter Fieldname	Field Value (h)	supports
major_version_number	0x0,...,0xA	MSB ₄ (certificate_version)
minor_version_number	0x0,...,0xA	LSB ₄ (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 encode in bslbf format. 16 Major and 16 Minor versions are supported. For 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.

number of certificate in chain	Value (h)	remark
0	0x0	will signal absence of <i>ri_certificate</i> e.g. 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 7 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 OCSF response chain.

number of responses in chain	Value (h)	remark
0	0x0	will signal absence of ocsp_response e.g. 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 7 OCSF 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 OCSF responses for the RI's certificate chain. The Device SHALL NOT fail due to the presence of more than one OCSF response element. A Device SHALL check that an OCSF response is present in the received message. If no OCSF 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 presence of the parameter it describes:

local_time_offset_flag	Value (h)	remark
data absent	0x0	
data present	0x1	

time_stamp_flag - Binary flag to signal presence of the parameter it describes:

time_stamp_flag	Value (h)	remark
data absent	0x0	
data present	0x1	

broadcast_group_key_flag - The flag expresses how many broadcast_group_keys (a.k.a. BGK) 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 9 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 broadcast_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.

broadcast_group_key_flag	Value (h)	remark
data absent	0x0	will signal absence of keyset_block e.g. on error status to save bandwidth.
reserved for future use	0x1-0x7	not used in this version of the specification
set of (8) broadcast_group_key	0x8	
set of (9) broadcast_group_key	0x9	
reserved for future use	0xA-0xF	not used in this version of the specification

signature_type_flag - A flag to signal type of signature algorithm used:

signature_type_flag	Value (h)	remark
RSA 1024	0x0	
RSA 2048	0x1	CMLA requirement (2004-2007)
RSA 4096	0x2	
reserved for future use	0x3	not used in this version of the specification

short_udn_flag - Binary flag to signal presence of the parameter it describes:

short_udn_flag	Value (h)	remark
data absent	0x0	
data present	0x1	

surplus_block_flag - Binary flag to signal the presence of the parameter it describes:

surplus_block_flag	Value (h)	remark
data absent	0x0	
data present	0x1	

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 - An symmetric AES encryption key to address a unique group. This key is also known as UGK. The key length SHALL be 128 bit.

Note: This key is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

broadcast_group_key - An (set of) AES symmetric encryption key(s) which are used for the zero message broadcast_group_key deduction of the key needed to decrypt the SEK and/or PEK. These broadcast_group_key is also known as BGK. The key length SHALL be 128 bit.

Note: This key is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

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 bit.

Note: This key is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

unique_device_filter - A [EUROCRYPT] style addressing scheme used to filter for messages like BCROs. A device address consists of 5 bytes and is unique within an operation. The shared address is defined as the 4 most significant bytes of the unique address. The least significant byte (byte 5) defines the position (0....255) in the group that shares an address. This means that each group consists of 256 members. An access mask, in an entitlement, is used to identify individual members. So if for a particular group only member 5 and 100 are allowed to have access to a service then their corresponding bits are set in the access mask. Take the device_id_mask equal to 252 (1111 1100_h) then the least significant byte of the device_id is masked and thereby creating a shared address. This address is also known as UDF.

Note: This address is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

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 bit.

Note: This key is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

local_domain_key - An AES symmetric key to address a unique device. This key is also known as LDK. The key length SHALL be 128 bit.

Note: This key is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

longform_domain_id() - This parameter is also known as the Longform Local Domain Filter (LLDF). Please refer to A.13.3 for the definition. The longform_domain_id() is used for mixed-mode operation. Note: This address is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

shortform_domain_id – This parameter is also known as the Shortform Local Domain Filter (SLDF). Please refer to A.13. An addressing scheme used to filter for messages like BCROs. The shortform_domain_id is used for broadcast mode of operation.

Note: This address is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

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. Refer to Appendix A.4 for calculation of the UTC and Modified Julian Date (MJD). This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit BCD.

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

local_time_offset - This parameter indicates the local time offset from the (UTC) drm_time as explained in Annex A.4.1.

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. (Note: please note that this parameter can also be used against replay attacks.)

registration_timestamp_end - Indicates from what time onwards the registration data expires. This is an extra mechanism above the expiration date of the RI certificate. (Note: please note that 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 A.11.

Note Message result:

The stored RI Context SHALL at a minimum contain:

- RI ID, Unique device filter (UDF).
- following keys:
 - UGK, BGK1..n and/or UDK
 - RIAK.
 - SK

If domain addressing via an OMA DRM 2.0 domain is required the keyset SHALL (additionally to the standard addressing above) include following keys:

- LDK.
- Shortform Local Domain Filter (SLDF). A.k.a. “shortform_domain_id”. Refer to A.13.1.
- For mixed-mode devices domain context SHALL additionally contain:
 - Longform Local Domain Filter (LLDF). A.k.a. “longform_domain_id()”. Refer to A.13.3.
- 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 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 an ESG 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 context for broadcast mode of operation.
- For standard addressing the keyset SHALL include a valid set of :
 - UGK, BGK1..n and/or UDK keys
 - RIAK key. A single RIAK key is bound to a single broadcast group (e.g. 256 or 512 members).
 - Unique device filter (UDF).
- If domain addressing via an OMA DRM 2.0 domain is required the keyset SHALL (additionally to the standard addressing above) include a valid set of :
 - LDK key.
 - Shortform Local Domain Filter (SLDF). A.k.a. “shortform_domain_id”. Refer to A.13.1.

And in case of mixed-mode operation devices the keyset SHALL contain:

- A Longform Local Domain Filter (LLDF, a.k.a. “longform_domain_id()”) that matches the SLDF. Refer to A.13.3.

6.4.3.7.2.2 Protection of the keyset

The device_registration_response() message is split in two parts: device specific (time bound) data and global (not time bound) data.

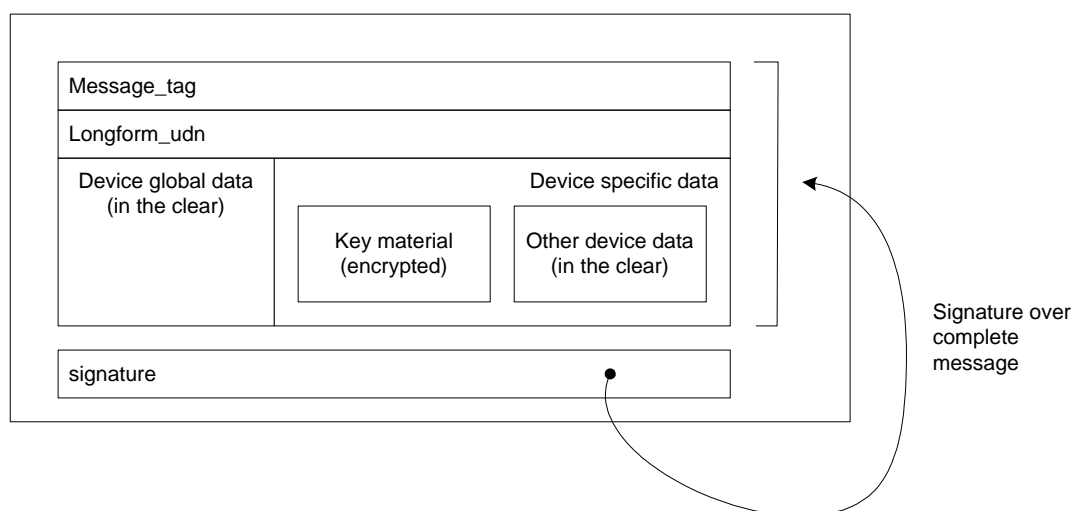


Figure 35: device_registration_response() message

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:

1. Generate a (128 or 192 or 256) bit AES key to be used as session key (SK) for the `device_registration_response()` message.
2. Concatenate the keyset (UGK, BGK1..n, UDK, RIAK, UDF and/or LDK, SLDF plus optional LLDF if applicable) under rules of [FIPS_197] and the Tag Length Format described in section A.13.
3. Encrypt the keyset using [AES_WRAP] using the generated SK as (AES-WRAP style) KEK. This will produce the `keyset_block`.
4. Calculate the part of the keyblock that would fit into the RSA block (depending on the size of RSA used, be that 1024, 2048 or 4096), including the SK and under implementation rules of the PKCS#1. If the `keyset_block` fits into one RSA block continue at step 5. Else continue at step 4.
5. 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()`.
6. 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]. This will produce the `sessionkey_block()`.
7. Concatenate the (non encrypted) parameters that were not used in the `key_block` and create the message "header" from this. Refer to 6.4.3.7.2.3 for details. (for reason of completeness: of course the `sessionkey_block()`, the (optional) `surplus_block()` and the `signature_block` are not part of the message header)
8. Concatenate the message "header" and the `sessionkey_block()` . If the SK plus `keyset_block` including 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 [PKCS#1]. Please refer to section A.11. This will produce the `signature_input_data`.
9. 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 A.11. This will produce the `signature_block`.
10. The `device_registration_response()` message comprises of the message "header" plus `sessionkey_block()`, optionally the `surplus_block()` and the `signature_block`.

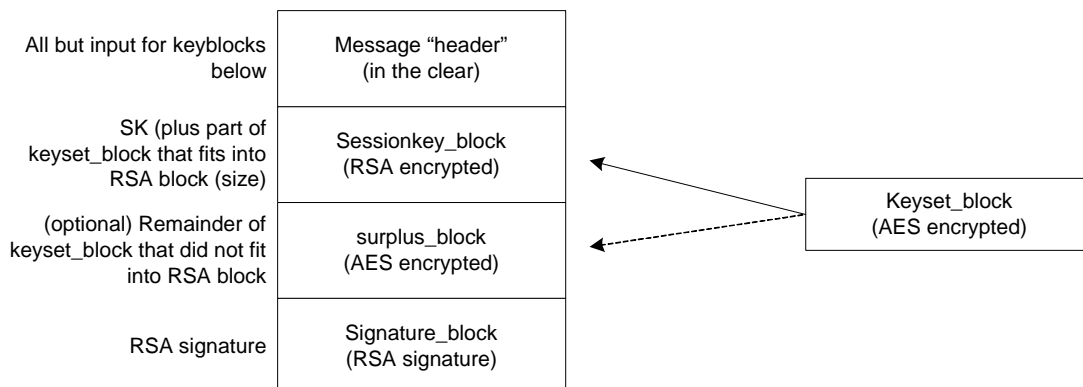


Figure 36: structure of `device_registration_response()` message.

Concluding: The number of RSA blocks used should be kept to a minimum. The AES `surplus_block()` is present if and 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(). Refer to appendix A.7 for calculations on the surplus_block_size.

Decryption of the encrypted message SHALL adhere to the following rules:

1. Locate the message via message_tag
2. Verify if the message is intended for this device by comparing the long_form_udn with the UDN stored in the device.
3. Verify the signature_block of the message by using the public key from the RI.
4. 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). Then locate the keyset_block part from the header and (eventual) padding (according to PKCS#1).
5. (Optionally) If there is a surplus_block() concatenate this part to the keyset_block. This will complete the keyset_block.
6. Use the SK to decrypt the keyset_block.
7. Allocate the individual keyset_items from the keyset_block according to [AES_WRAP] and the Tag Length Format described in section A.13.

Note: 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.

6.4.3.7.2.3 Message syntax

Table 40: message syntax

fields	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
broadcast_group_key_flag	4	bslbf
short_udn_flag	1	bslbf
signature_type_flag	2	bslbf
surplus_block_flag	1	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
}		
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 (short_udn_flag == 0x1) {		
short_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()	(*1)	bslbf
padding_bits	(*2)	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
}		
}		

key:

(*1) for details please refer to appendix A.7

(*2) (surplus_block() length) mod 8

6.4.3.7.3 (Force to) Re-register - re_register_msg() message

6.4.3.7.3.1 Message description

Using the 1-pass IRD protocol (refer to 6.4.3.4) the RI sends a register_msg message, indirectly triggering a (re)registration . The message is specified as follows:

Table 41: message description

re_register_msg()		
Parameter name	(M)andatory / (O)ptional ⁶	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	

message_tag - This parameter identifies the type of the message. Refer to section A.14 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 doesn't support. The value of the protocol_version of this message is set to 0x0 (i.e. the original format).

If set to 0x0 the format specified in the this version of the specification is used. If set to anything else than 0x0, then the format is beyond the scope of this version of the specification.

longform_udn() - The long form of the UDN. Refer to section 6.4.3.6 for details.

Table 42: 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 interaction 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 back to the OOB re-registration dialogue. Please note that 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. Please note that a mixed mode device will remain to have full interactive channel capabilities after receiving this status code, but will not use the interactive channel.

signature_type_flag - A flag to signal type of signature algorithm used:

⁶ key: (O)ptional means that the user of the message MAY include the parameter in the message, but the device MUST support the interpretation of the parameter. (M)andatory means that the user of the message SHALL include the parameter in the message.

signature_type_flag	Value (h)	remark
RSA 1024	0x0	
RSA 2048	0x1	CMLA requirement (2004-2007)
RSA 4096	0x2	
reserved for future use	0x3	not used in this version of the specification

certificate_version - is a numerical representation of the version of the RI certificate. Using the certificate_version parameter the device can decide if it is needed to update the RI certificate (if it was stored before).

Table 43: description of certificate_version parameter

Parameter Fieldname	Field Value (h)	supports
major_version_number	0x0,...,0xA	MSB ₄ (certificate_version)
minor_version_number	0x0,...,0xA	LSB ₄ (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 encode in bslbf format. 16 Major and 16 Minor versions are supported. For 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.

number of certificate in chain	Value (h)	remark
0	0x0	will signal absence of ri_certificate e.g. 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 7 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.

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.

ocsp_response_counter - This parameter indicates the depth of the OCSP response chain.

number of responses in chain	Value (h)	remark
0	0x0	will signal absence of ocsp_response e.g. 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 7 OCSP responses.

r_length - This parameter indicates the length in bytes of the obsp_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 A.11.

6.4.3.7.3.2 Message syntax

Table 44: message syntax

fields	length	Type
re_register_msg() {		
/* signature protected part starts here */		
message_tag	8	bslbf
protocol_version	4	bslbf
longform_udn()	80	bslbf
flags {		
signature_type_flag	1	bslbf
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
reserved for future use	5	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
}		
}		

6.4.3.7.4 Update RI certificate - update_ri_certificate_msg() message

Using the 1-pass IRD protocol (refer to 6.4.3.4) the RI sends a 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 described in section 6.4.3.7.3, with the only adaptation being that the message_tag is different. Refer to section A.14 for the value of the message_tag.

6.4.3.7.5 Updating the DRM time - update_drmtime_msg() message

6.4.3.7.5.1 Message description

Using the 1-pass IRD protocol (refer to 6.4.3.4) the RI sends a update_drmtime trigger message with the drmtime to the device as specified below:

Table 45: message description

update_drmtime_msg()		
Parameter name	(M)andatory / (O)ptional ⁷	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	

message_tag - This parameter identifies the type of the message. Refer to section A.14 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 doesn't support. The value of the protocol_version of this message is set to 0x0 (i.e. the original format).

If set to 0x0 the format specified in the this version of the specification is used. If set to anything else than 0x0, then the format is beyond the scope of this version of the specification.

status - The status parameter SHALL indicate one of the values explained in the following table. The device SHALL ignore messages with other error values.

Table 46: 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 it's time. As an extra result the device will determine the eventual clock drift and notify this to the RI per ARC (offline notification of short device data; refer to section 6.4.4.3. Please note: this capability should be used with great care.)

local_time_offset_flag - Binary flag to signal presence of the parameter it describes:

local_time_offset_flag	Value (h)	remark
data absent	0x0	
data present	0x1	

signature_type_flag - A flag to signal type of signature algorithm used:

signature_type_flag	Value (h)	remark
RSA 1024	0x0	
RSA 2048	0x1	CMLA requirement (2004-2007)
RSA 4096	0x2	
reserved for future use	0x3	not used in this version of the specification

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. Refer to Appendix A.4 for calculation of the UTC and Modified Julian Date (MJD). This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit BCD.

⁷ key: (O)ptional means that the user of the message MAY include the parameter in the message, but the device MUST support the interpretation of the parameter. (M)andatory means that the user of the message SHALL include the parameter in the message.

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

local_time_offset - This parameter indicates the local time offset from the (UTC) `drm_time` as explained in Annex A.4.1.

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 A.11.

6.4.3.7.5.2 Message syntax

Table 47: message syntax

fields	length	type
<code>update_drmtime_msg(){</code>		
<code>/* signature protected part starts here */</code>		
<code>message_tag</code>	8	bslbf
<code>protocol_version</code>	4	bslbf
<code>reserved_for_future_use</code>	4	bslbf
<code>status</code>	8	bslbf
<code>flags {</code>		
<code> local_time_offset_flag</code>	1	bslbf
<code> signature_type_flag</code>	1	bslbf
<code> reserved for future use</code>	6	bslbf
<code>}</code>		
<code>drm_time</code>	40	mjdutc
<code>if (local_time_offset_flag == 0x1) {</code>		
<code> local_time_offset</code>	16	bslbf
<code>}</code>		
<code>/* signature protected part ends here */</code>		
<code>if (signature_type_flag == 0x0){</code>		
<code> signature_block</code>	1024	bslbf
<code>} else if (signature_type_flag == 0x1)</code>		
<code> signature_block</code>	2048	bslbf
<code>} else if (signature_type_flag == 0x2)</code>		
<code> signature_block</code>	4096	bslbf
<code>}</code>		
<code>}</code>		

6.4.3.7.6 Update the contact number – `update_contact_number_msg()` message

6.4.3.7.6.1 Message description

Using the 1-pass IRD protocol (refer to 6.4.3.4) the RI sends a `update_contact_number_msg()` message with a (set of) contact number(s) to the device as specified below:

Table 48: message description

update_contact_number_msg()		
Parameter name	(M)andatory / (O)ptional ⁸	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	

message_tag - This parameter identifies the type of the message. Refer to section A.14 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 doesn't support. The value of the protocol_version of this message is set to 0x0 (i.e. the original format).

If set to 0x0 the format specified in the this version of the specification is used. If set to anything else than 0x0, then the format is beyond the scope of this version of the specification.

status - The status parameter SHALL indicate one of the values explained in the following table. The device SHALL ignore messages with other error values.

Table 49: 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 (e.g. via ESG).

signature_type_flag - A flag to signal type of signature algorithm used:

signature_type_flag	Value (h)	remark
RSA 1024	0x0	
RSA 2048	0x1	CMLA requirement (2004-2007)
RSA 4096	0x2	
reserved for future use	0x3	not used in this version of the specification

certificate_version - is a numerical representation of the version of the RI certificate. Using the certificate_version parameter the device can decide if it is needed to update the RI certificate (if it was stored before).

⁸ key: (O)ptional means that the user of the message MAY include the parameter in the message, but the device MUST support the interpretation of the parameter. (M)andatory means that the user of the message SHALL include the parameter in the message.

Table 50: description of certificate_version parameter

Parameter Fieldname	Field Value (h)	supports
major_version_number	0x0,...,0xA	MSB ₄ (certificate_version)
minor_version_number	0x0,...,0xA	LSB ₄ (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 encode in bslbf format. 16 Major and 16 Minor versions are supported. For 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.

number of certificate in chain	Value (h)	remark
0	0x0	will signal absence of ri_certificate e.g. 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 7 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.

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.

ocsp_response_counter - This parameter indicates the depth of the OCSP response chain.

number of responses in chain	Value (h)	remark
0	0x0	will signal absence of ocsp_response e.g. 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 7 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.

contacts_counter - This parameter indicates the number of contacts carried in the message.

contact – This object specifies the contact. Please refer to 6.4.3.7.6.2.1.

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 A.11.

6.4.3.7.6.2 Message syntax

Table 51: message syntax

fields	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
signature_type_flag	1	bslbf
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
reserved for future use	5	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()		
}		
/* 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
}		
}		

6.4.3.7.6.2.1 Format of the contact object

Table 52: contact object format

Field	length	type
contact(){		
contact_type	4	uimsbf
reserved for future use	4	bslbf
contact_length	8	uimsbf
contactdata	8*contact_length	bslbf
}		

contact_type: This field specifies the type of action as listed in table 53.

Table 53: 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
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. Maximum length of the contacts is specified in table 53.

UTF-8 [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).

For example: a URL is limited to 30 characters. The 30 URL UTF-8 characters are translated into bytes as follows:

E.g.: "Western" languages - character is 1 byte - Longest URL encoded as bytes is 1×30 characters = 30 bytes.

E.g.: 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.

contact types	contactdata encoding rules
phone numbers	The phone number is encoded as alphabetic, supporting telephone numbers like: "0800-123456789" but also for example: "0800-philips". The string that forms the phone number is encoded using UTF-8.
SMS numbers	The SMS number is encoded as hexadecimal, supporting telephone numbers like: "0800-123456789" but also for example: "philips+subscribe". The string that forms the SMS number is encoded using UTF-8.
URLs	The URL is encoded as hexadecimal, according to [RFC 1738], supporting URLs like: www.philips.com/start . The string that forms the URL is encoded using UTF-8.

6.4.4 Domain joining and leaving

Interactive devices will adhere to [OMA_DRM_DRM].

- Interactive devices will therefore use OMA DRM 2.0 domain ID.

Broadcast devices will adhere to the mechanisms as described in this section.

- Broadcast devices will use “shortform_domain_id” a.k.a. SLDF.

Mixed-mode SHALL have the "interoperability" requirement to support both domain ID formats of interactive and broadcast devices:

- Mixed-mode device will receive:
 - “longform_domain_id()”, a.k.a. LLDF, which is a translation of OMA DRM 2.0 domain ID.
 - “shortform_domain_id” a.k.a. SLDF.
- 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.

6.4.4.1 protocol overview

The theory of operation (refer to section 5.3) results in the specification of several protocols:

- offline protocols (from device to RI)

protocol	section	purpose
offline Domain Join Request protocol	6.4.4.2	request to join a domain
offline Domain Leave Request protocol	6.4.4.3	request to leave a domain

- 1-pass protocols (from RI to device)

protocol	section	purpose
1-pass binary Push Device Registration protocol	6.4.3.4	transmit registration data to device
1-pass binary Inform Registered Device protocol	6.4.3.5	inform device via messages.

The protocols interrelate in following way (roundtrip):

kicking off action...	...results in
offline domain join request. (request to join a domain).	domain_registration_response() message. (transmit registration data to device).
offline domain leave request (request to leave a domain)	domain_update_response() message. (inform device via messages)
join_domain_msg() (inform device via messages)	offline domain join request, which on it's turn may result in domain_registration_response() as listed above.
leave_domain_msg() (inform device via messages)	offline domain leave request, which on it's turn may result in domain_update_response() as listed above.

6.4.4.2 offline 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. (refer to section 6.4.3.3).

6.4.4.3 offline 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. (refer to section 6.4.3.3).

6.4.4.4 Binary messages

6.4.4.4.1 Domain data - domain_registration_response() message

6.4.4.4.1.1 Message description

Using the 1-pass PDR protocol (refer to 6.4.3.4) the RI sends a domain_registration_response() message, informing the device of a new domain keyset. The message is specified below:

Table 54: message description

domain_registration_response()		
Parameter name	(M)andatory / (O)ptional ⁹	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
local_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

message_tag - This parameter identifies the type of the message. Refer to section A.14 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 doesn't support. The value of the protocol_version of this message is set to 0x0 (i.e. the original format).

If set to 0x0 the format specified in the this version of the specification is used. If set to anything else than 0x0, then the format is beyond the scope of this version of the specification.

longform_udn() - The long form of the UDN. Refer to section 6.4.3.6 for details.

status - The status parameter SHALL indicate one of the values explained in the following table. The device SHALL ignore messages with other error values.

⁹ key: (O)ptional means that the user of the message MAY include the parameter in the message, but the device MUST support the interpretation of the parameter. (M)andatory means that the user of the message SHALL include the parameter in the message.

Table 55: 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.

device_nonce - The device_nonce is the nonce which was present in the request (using the offline NSD protocol) to which this message is a response. This nonce is encoded in BCD.

time_stamp_flag - Binary flag to signal presence of the parameter it describes:

time_stamp_flag	Value (h)	remark
data absent	0x0	
data present	0x1	

certificate_version - is a numerical representation of the version of the RI certificate. Using the certificate_version parameter the device can decide if it is needed to update the RI certificate (if it was stored before).

Table 56: description of certificate_version parameter

Parameter Fieldname	Field Value (h)	supports
major_version_number	0x0,...,0xA	MSB ₄ (certificate_version)
minor_version_number	0x0,...,0xA	LSB ₄ (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 encode in bslbf format. 16 Major and 16 Minor versions are supported. For 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.

number of certificate in chain	Value (h)	remark
0	0x0	will signal absence of ri_certificate e.g. 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 7 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 OCSF 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 OCSF response chain.

number of responses in chain	Value (h)	remark
0	0x0	will signal absence of ocsp_response e.g. 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 7 OCSF 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 OCSF responses for the RI's certificate chain. The Device SHALL NOT fail due to the presence of more than one OCSF response element. A Device SHALL check that an OCSF response is present in the received message. If no OCSF 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. (Note: please note that 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. (Note: please note that this parameter can also be used against replay attacks.)

signature_type_flag - A flag to signal type of signature algorithm used:

signature_type_flag	Value (h)	remark
RSA 1024	0x0	
RSA 2048	0x1	CMLA requirement (2004-2007)
RSA 4096	0x2	
reserved for future use	0x3	not used in this version of the specification

keyset_block_length - This parameter indicates the length in bits of the total keyset_block. That is the part in the sessionkey_block().

local_domain_key - An AES symmetric key to address a unique device. This key is also known as LDK. The key length SHALL be 128 bit.

Note: This key is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

longform_domain_id() – This parameter is also known as the Longform Local Domain Filter (LLDF). Please refer to A.13.3 for the definition. The longform_domain_id() is used for mixed-mode operation. Note: This address is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

shortform_domain_id – This parameter is also known as the Shortform Local Domain Filter (SLDF). Please refer to A.13. An addressing scheme used to filter for messages like BCROs. The shortform_domain_id is used for broadcast mode of operation.

Note: This address is wrapped into the keyset_block. (Refer to 6.4.3.7.2.2).

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 A.11.

Note Message result:

The stored domain context SHALL at a minimum contain:

- Following keys:
 - LDK.
 - Shortform Local Domain Filter (SLDF). A.k.a. “shortform_domain_id”. Refer to A.13.1.
- For mixed-mode operation, devices’ domain context SHALL additionally contain:
 - Longform Local Domain Filter (LLDF). A.k.a. “longform_domain_id()”. Refer to A.13.3.
- 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 an ESG 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 explained in 6.4.3.7.2.2) include a valid set of :
 - LDK key.
 - Shortform Local Domain Filter (SLDF). A.k.a. “shortform_domain_id”. Refer to A.13.1.

And in case of mixed-mode operation devices the keyset SHALL contain:

- A Longform Local Domain Filter (LLDF, a.k.a. “longform_domain_id()”) that matches the SLDF. Refer to A.13.3.

6.4.4.4.1.2 Protection of the keyset

The domain_registration_response() message is split in two parts: device specific (time bound) data and global (not time bound) data.

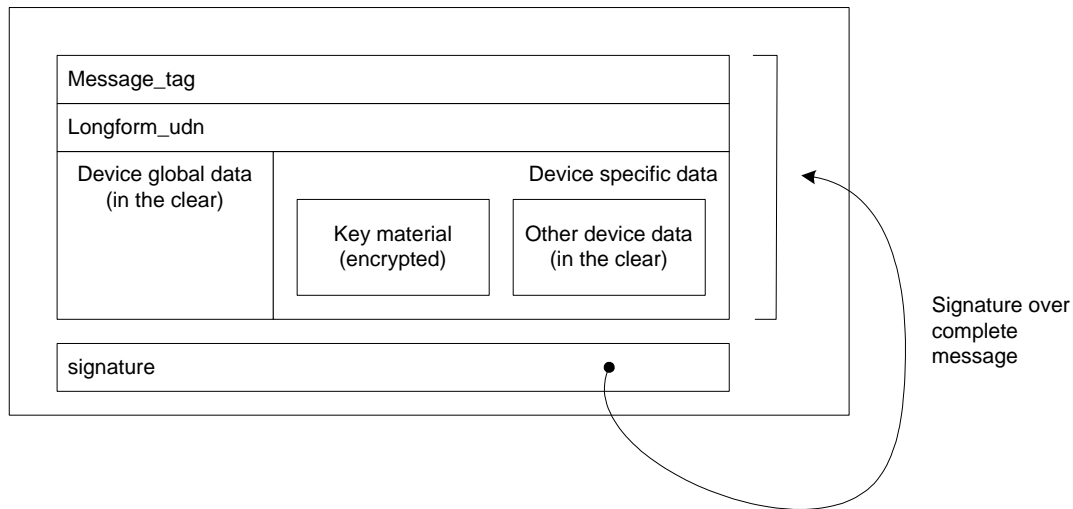


Figure 37: domain_registration_response() message

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:

1. Generate a (128 or 192 or 256) bit AES key to be used as session key (SK) for the `domain_registration_response()` message.
2. Concatenate the keyset (LDK, SLDF plus optional LLDF if applicable) under rules of [FIPS_197] and the Tag Length Format described in section A.13.
3. Encrypt the keyset using [AES_WRAP] using the generated SK as (AES-WRAP style) KEK. This will produce the *keyset_block*.
4. Calculate the part of the keyblock that would fit into the RSA block (depending on the size of RSA used, be that 1024, 2048 or 4096), including the SK and under implementation rules of the PKCS#1.
5. 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]. This will produce the *sessionkey_block()*.
6. Concatenate the (non encrypted) parameters that were not used in the *key_block* and create the message "header" from this. Refer to 6.4.4.4.1 for details. (for reason of completeness: of course the *sessionkey_block()* and the *signature_block* are not part of the message header)
7. Concatenate the message "header" and the *sessionkey_block()* . Hash the result under implementation guidelines of [PKCS#1]. Please refer to section A.11. This will produce the *signature_input_data*.
8. 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 A.11. This will produce the *signature_block*.
9. The `domain_registration_response()` message comprises of the message "header" plus *sessionkey_block()* and the *signature_block*.

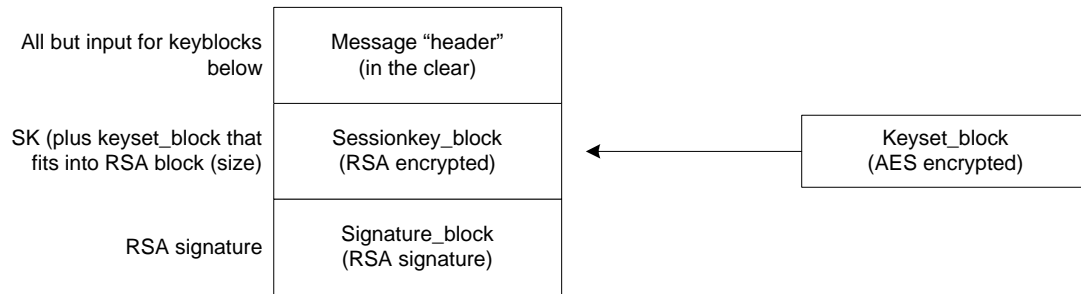


Figure 38: structure of domain_registration_response() message.

Concluding: The number of RSA blocks used should be kept to a minimum. The AES surplus_block() is present if and 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(). Refer to appendix A.7 for calculations on the surplus_block_size.

Decryption of the encrypted message SHALL adhere to the following rules:

1. Locate the message via `message_tag`
2. Verify if the message is intended for this device by comparing the `long_form_udn` with the UDN stored in the device.
3. Verify the `signature_block` of the message by using the public key from the RI.
4. 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). Then locate the `keyset_block` part from the header and (eventual) padding (according to PKCS#1).
5. Use the SK to decrypt the `keyset_block`.
6. Allocate the individual `keyset_items` from the `keyset_block` according to [AES_WRAP] and the Tag Length Format described in section A.13.

Note: 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.

6.4.4.4.1.3 Message syntax

Table 57: message syntax

fields	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)		
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
}		
}		

6.4.4.4.2 Updating a domain - domain_update_response() message

6.4.4.4.2.1 Message description

Using the 1-pass IRD protocol (refer to 6.4.3.5) the RI sends a domain_update_response() message, informing the device that it left a particular domain. The message is specified below:

Table 58: message description

domain_update_response()		
Parameter name	(M)andatory / (O)ptional ¹⁰	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

message_tag - This parameter identifies the type of the message. Refer to section A.14 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 doesn't support. The value of the protocol_version of this message is set to 0x0 (i.e. the original format).

If set to 0x0 the format specified in the this version of the specification is used. If set to anything else than 0x0, then the format is beyond the scope of this version of the specification.

longform_udn() - The long form of the UDN. Refer to section 6.4.3.6 for details.

status - The status parameter SHALL indicate one of the values explained in the following table. The device SHALL ignore messages with other error values.

Table 59: Status values

status value	meaning
Success	The message informs the device that the RI has removed this device from the domain it was registered in. 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

device_nonce - The device_nonce is the nonce which was present in the request (using the offline NSD protocol) to which this message is a response. This nonce is encoded in BCD.

certificate_version - is a numerical representation of the version of the RI certificate. Using the certificate_version parameter the customer device can decide if it is needed to update the RI certificate (if it was stored before).

¹⁰ key: (O)ptional means that the user of the message MAY include the parameter in the message, but the device MUST support the interpretation of the parameter. (M)andatory means that the user of the message SHALL include the parameter in the message.

Table 60: description of certificate_version parameter

Parameter Fieldname	Field Value (h)	supports
major_version_number	0x0,...,0xA	MSB ₄ (certificate_version)
minor_version_number	0x0,...,0xA	LSB ₄ (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 encode in bslbf format. 16 Major and 16 Minor versions are supported. For 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.

number of certificate in chain	Value (h)	remark
0	0x0	will signal absence of ri_certificate e.g. 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 7 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.

number of responses in chain	Value (h)	remark
0	0x0	will signal absence of obsp_response e.g. 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 7 OSCP responses.

r_length - This parameter indicates the length in bytes of the obsp_response.

ocsp_response() - This parameter, when present, SHALL be a complete set of valid OSCP responses for the RI's certificate chain. The Device SHALL NOT fail due to the presence of more than one OSCP response element. A Device SHALL check that an OSCP response is present in the received message. If no OSCP 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 SLDF.

signature_type_flag - A flag to signal type of signature algorithm used:

signature_type_flag	Value (h)	remark
RSA 1024	0x0	
RSA 2048	0x1	CMLA requirement (2004-2007)
RSA 4096	0x2	
reserved for future use	0x3	not used in this version of the specification

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 A.11.

6.4.4.4.2.2 Message syntax

Table 61: message syntax

fields	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
}		
}		

6.4.4.4.3 (Force to) Join a domain - join_domain_msg() message

Using the 1-pass IRD protocol (refer to 6.4.3.5) 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 described in section 6.4.3.7.3, with the only adaptation being that the message_tag is different. Refer to section A.14 for the value of the message_tag.

6.4.4.4.6 (Force to) Leave a domain - leave_domain_msg() message

Using the 1-pass IRD protocol (refer to 6.4.3.5) 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 described in section 6.4.3.7.3, with the only adaptations being that :

- the message_tag is different. Refer to section A.14 for the value of the message_tag.
- the shortform_domain_id is incorporated, which is the SLDF.

For the message **description** with an explanation of the parameters refer to the re_register_msg() message. For sake of completion the complete leave_domain_msg() message **syntax** is explained below:

6.4.4.4.6.1 message syntax

Table 62: message syntax

fields	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	1	bslbf
ri_certificate_counter	3	bslbf
ocsp_response_counter	3	bslbf
reserved for future use	1	bslbf
}		
shortform_domain_id	48	uimsbf
certificate_version	8	bslbf
for(cnt1=0; cnt1 < ri_certificate_counter ;cnt1++){		
c_length	16	uimsbf
ri_certificate(0	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

}		
}		

6.4.5 Token handling

6.4.5.1 Protocol overview

The theory of operation (refer to section A.16) results in the specification of several protocols:

- offline protocols (from device to RI)

protocol	section	purpose
token request protocol	6.4.5.2	request to purchase tokens
token reporting protocol	6.4.5.3	protocol to report the consumption of tokens

- 1-pass protocols (from RI to device)

protocol	section	purpose
1-pass binary Push Device Registration protocol	6.4.3.4	transmit registration data to device
1-pass binary Inform Registered Device protocol	6.4.3.5	inform device via messages.

The protocols interrelate in following way (roundtrip):

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)

6.4.5.2 token request protocol

When the user of a device wants to obtain tokens, he uses the NSD protocol with the token_request action type. (refer to section 6.4.3.3).

6.4.5.3 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. (refer to section 6.4.3.3.).

6.4.5.4 Binary messages

6.4.5.4.1 delivering tokens – token_delivery_response() message

6.4.5.4.1.1 Message description

Using the 1-pass PDR protocol (refer to section 6.4.3.4) the RI sends a token_delivery_response() message, informing the device of the delivery of new tokens. The message is specified below:

Table 63: token delivery response message description

token_delivery_response()		
Parameter name	(M)andatory / (O)ptional ¹¹	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 version of the specification	not encrypted
position_in_group	M	not encrypted
domain_id_extension	not used in this version of the specification	not encrypted
domain_generation	not used in this version of the specification	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

message_tag - This parameter identifies the type of the message. Refer to section A.14 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 doesn't support. The value of the protocol_version of this message is set to 0x0 (i.e. the original format).

If set to 0x0 the format specified in the this version of the specification is used. If set to anything else than 0x0, then the format is beyond the scope of this version of the specification.

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 - 3-bit field indicating the addressing mode used by this message. Table 64 lists all possible address modes. Not that not all modes are used in this version of the specification for the token delivery response message.

¹¹ key: (O)ptional means that the user of the message MAY include the parameter in the message, but the device MUST support the interpretation of the parameter. (M)andatory means that the user of the message SHALL include the parameter in the message.

Table 64 address_mode for token delivery response message

address_mode	description
0x0	addressing whole of unique group This addressing mode is not used in this version of the specification for the token delivery response message
0x1	addressing of broadcast 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 version of the specification 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 version of the specification for the token delivery response message
0x5-0x7	reserved for future use

one - 1-bit flag which SHALL have the value 0x1 in this version of the specification. This field MAY have value 0x0 in future versions of the specification

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 this message is addressed to a unique device in a group then the LSB of the address field is the MSB of the group position.

bit_access_mask - This field is not used in this version of the specification and MAY be used in future versions. It is indicated here, so devices according to this version of the specification 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 then this field specifies the position of the unique device in the given broadcast group. If group_size_flag is 0 then the position in the group is directly given by the position_in_group field. If group_size_flag is 1 then 9 bit 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 MSB. 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 version of the specification and MAY be used in future versions. It is indicated here, so devices according to this version of the specification 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 version of the specification and MAY be used in future versions. It is indicated here, so devices according to this version of the specification 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 X509PKISHash in [OMA-DRM-DRM].

status - The status parameter SHALL indicate one of the values explained in the following table. The device SHALL ignore messages with other error values.

Table 65: 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	<p>The RI did receive a token consumption message, but that it was erroneous and that the device should redo the last token consumption message.</p> <p>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.</p>
NoTokenConsumptionMessage	<p>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.</p> <p>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.</p>

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 which are still not consumed, without ever having to report their consumption.

earliest_reporting_time_flag - Binary flag to signal presence of the parameter it describes:

earliest_reporting_time field	Value (h) of earliest_reporting_time_flag	remark
data absent	0x0	
data present	0x1	

latest_reporting_time_flag - Binary flag to signal presence of the parameter it describes:

latest_reporting_time field	Value (h) of latest_reporting_time_flag	remark
data absent	0x0	
data present	0x1	

token_quantity_flag - Binary flag to signal presence of the parameter it describes:

token_quantity field	Value (h) of token_quantity_flag	remark
data absent	0x0	
data present	0x1	

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.

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.

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.

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 the next table.

Table 66 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 version of the specification for the token delivery response message
0x1 (broadcast group)	This addressing mode is not used in this version of the specification for the token delivery response message
0x2-0x3 (unique device)	Token Delivery Key
0x4 (domain)	This addressing mode is not used in this version of the specification 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 the next table.

Table 67 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 version of the specification for the token delivery response message
0x1 (broadcast group)	This addressing mode is not used in this version of the specification for the token delivery response message
0x2-0x3 (unique device)	Token Delivery Key
0x4 (domain)	This addressing mode is not used in this version of the specification 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 [RFC 2104]). The MAC is used for integrity check of this message. The key used to create the MAC is the token delivery response message authentication key TDRMAK as defined in Annex A.9.5. Devices SHALL NOT use token delivery response messages with an invalid MAC.

Note Message result:

- More information on device actions after the reception of this message can be found in section A.16.2.

6.4.5.4.1.3 Message syntax

Table 68: token delivery response message syntax

fields	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){ ¹²		
bit_access_mask	256	bslbf
}else if(address_mode == 0x1 && group_size_flag == 1){		
bit_access_mask ¹²	512	bslbf
}else if (address_mode & 0x6 == 0x2){		
position_in_group	8	uimsbf
}else if (address_mode == 0x4){ ¹²		
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) {		

¹² Although this addressing mode is indicated here to facilitate future upgrades, this addressing mode is NOT used in this version of the specification. for the token delivery response message

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){		
encrypted_token_quantity	32	bslbf
}		
encrypted_report_authentication_key	128	bslbf
/* encrypted part ends here		
/* MAC protected part ends here */		
MAC	96	bslbf
}		

Note that all reserved for future use fields SHALL have the value 0 for token delivery response messages created according to this version of the specification.

7 Rights Issuer Services

[Note to the editor: This chapter sets out a number of requirements on the ESG and must be checked once the ESG specification is complete and available.]

Rights Issuer Streams are used to carry Registration Layer and Rights Management Layer objects and messages. These include all the messages that are allocated a message tag in chapter A.14.

Within this chapter, the objects and messages to be carried are referred to as “objects”.

The data carried by IP Datacast systems is logically divided into services. Each Rights Issuer Service consists of one or more IP streams.

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. The following types of RI Stream are described:

- Ad-hoc RI Stream
- Scheduled RI Stream
- In-Band RI Stream

Additionally, Rights Issuers MAY use Rights Issuer Streams to deliver messages in any way they require. 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.

An informative schedule MAY be broadcast for RI Services. Where available, this SHALL be provided as part of the ESG. It is used to indicate times at which data for particular sets of devices or Broadcast 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.

This specification 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 minimise 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 chapter 7.1.

7.1 Expected Mode of Operation (Informative)

It is foreseen that the system will be used in the following way. However, it is noted that considerable variation in actual operation is possible within the scope of this specification, in order to support the needs of Service Operation Centres and Rights Issuers. Any message can be carried in any RI Service, at the discretion of the Service Operation Centre and Rights Issuer.

- Using the ESG, a device can determine which Rights Issuer Service will be used to deliver messages to it. This service will be used to deliver the messages mentioned above.
- An RI Service can contain a number of streams, some of which carry scheduled data while others carry ad-hoc data. When a device is receiving an RI Service, it will receive all the streams within that service.
- A Scheduled RI Stream carries all the messages that an RI wishes to make available.

- These messages are grouped in time by device or Broadcast Group. A schedule giving the times at which information for particular sets of devices or Broadcast Groups will be carried is made available in the ESG. Devices need only listen to the service at the times relevant to it.
 - It is expected that all BCROs required by any authorised device to receive a protected service will be carried in a carousel format within a Scheduled RI Stream.
 - Future BCROs will also be carried, to prevent breaks in service when services keys are changed.
 - It is also expected that re-registration messages, domain update messages, etc will be carried.
 - RI Certificate Chain updates can be separately scheduled within the ESG. The mechanism specified in this chapter makes it possible for RIs to make these updates available in a scheduled stream alongside other messages, or for a stream to be available which continuously repeats the RI Certificate Chain message.
 - The bandwidth used for individual RI Services can be varied, for example to use any spare capacity that is available at certain times of the day.
- An Ad-hoc RI Stream is used to deliver messages with low latency. In order to receive an Ad-hoc RI Stream, a device will select the relevant RI Service – to do this it could be put into a special mode or have a particular service selected by the user. Examples of messages expected to be carried in an ad-hoc service include:
 - Registration messages, sent directly after a user has registered.
 - BCROs for services that a user has just purchased.
 - Domain control messages.
 - Token delivery messages.
 - Additionally, there can also be In-Band RI Streams. These are broadcast as a separate IP stream within, most likely, a media service. These services can be used in whatever way an RI requires, but it is expected that they will carry:
 - BCROs which require immediate delivery, probably to many devices. Examples include BCROs for Free To View services or for free previews.
 - BCROs for content items being delivered within the service.
 - Any message that an RI wishes to make available immediately.

7.2 Scheduled RI Stream

In a Scheduled RI Stream, the timing of message broadcast may be scheduled in some way, according to device or Broadcast 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 Broadcast Groups. Where provided, it SHALL be available in the ESG.

Note that 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.

7.3 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.

7.4 In-Band RI Streams within a Media Service

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 contain:

- Messages that need to be delivered immediately to large numbers of devices. Examples include Rights Objects for free previews 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 ESG.

7.5 Broadcast Format of RI Streams

All the objects defined in this specification are carried in Rights Issuer Streams. The format of these streams is defined in this chapter.

These streams SHOULD have the following characteristics:

- The bandwidth overhead of the stream format SHOULD be minimised.
- Objects of varying sizes (smaller than, similar to and larger than the size of an IP packet) SHOULD be efficiently carried.
- Devices SHOULD be able to start interpreting the stream at any packet.
- Where packet reception is unreliable or where packets have been reordered, devices SHOULD be able to determine which objects have been correctly and completely received.

Note that 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.

7.5.1 IP Characteristics

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 [RFC 768]. This specification does not specify any limits to the length of these IP packets – this will instead be determined by the underlying network.

Chapter 7.5.2 defines the format of the packets of the RI Stream.

7.5.2 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.

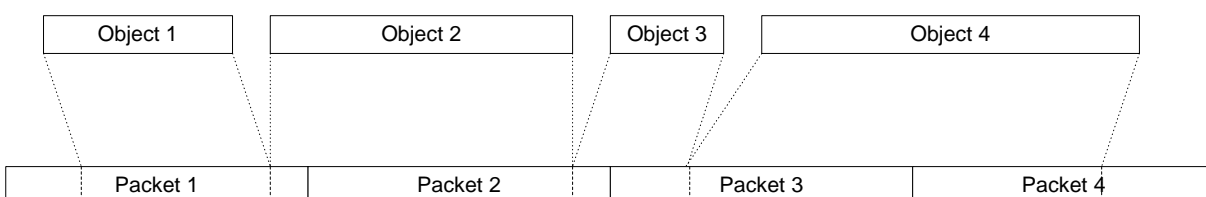


Figure 39: Example mapping of objects to RI Stream Packets

Objects are placed into packets according to the following rules:

- 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 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. Note that fragments will 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 as follows.

Table 69: Format of the Rights Issuer Stream

fields	length	format
<code>while(bytes left in packet){</code>		
<code>split_flag</code>	1	bslbf
<code>if(split_flag == 1) {</code>		
<code>object_id</code>	7	bslbf
<code>fragment_number_within_object</code>	4	bslbf
<code>total_number_of_fragments_for_object</code>	4	bslbf
<code>if(fragment_number_within_object ==</code> <code>total_number_of_fragments_for_object) {</code>		
<code>reserved_for_future_use</code>	4	
<code>remaining_length_in_packet</code>	12	bslbf
<code>}</code>		
<code>bytes_of_object()</code>		
<code>}</code>		
<code>else{</code>		
<code>reserved_for_future_use</code>	3	
<code>length_of_object</code>	12	bslbf
<code>bytes_of_object()</code>		
<code>}</code>		
<code>}</code>		

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.

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.

7.5.3 Implementation Notes

7.5.3.1 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 see chapter 7.5.3.2 as apparently missing packets could later be received due to packet reordering.

7.5.3.2 Changes in Packet Order (Informative)

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 or where this service protection scheme is used over other types of link.

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 later received 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 contained in all three packets. In order to implement this reconstruction scheme, the device buffers partly received objects for some time, and then reconstruct 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 implementation of any scheme of this kind is not required by this specification.

It is recommended that Service Operations Centres and Rights Issuers minimise changes in packet order within their systems.

7.5.3.3 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.

7.6 Mapping of Messages to RI Services and Streams

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 chapters describe how messages are mapped to services and streams.

7.6.1 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.

7.6.2 Rights Issuer Services Without Complete Schedule Information

If a Rights 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.

7.7 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:

- 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 version of the specification 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.

7.8 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 annex A.1. 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.

- 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 described in this chapter, 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.

7.9 Resending of BCROs

There is no guarantee that a device will receive the BCROs sent to it via the broadcast channel. A device may request that the BCROs be sent once again by the Rights Issuer.

7.9.1 Resending of BCROs to Interactive Devices

For an interactive device, requests to resend BCROs can be made via the interactivity channel. If the BCROs are to be delivered via the 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 one hour, 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 the broadcast channel, it MAY resend the BCROs for that device.

7.9.2 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 chapter 6.4.3.3. 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 one hour, 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.

7.10 Summary of Requirements for Rights Issuers

If a Rights Issuer delivers messages to devices via the 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 the broadcast channel, it will not have Rights Issuer Services and Streams, and the remainder of this chapter does not apply.

- Each Rights Issuer SHALL either:
 - Provide a complete schedule for their Rights Issues 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:
 - MAY contain any number of Scheduled or Ad-hoc RI Streams.

- 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 which it is entitled to receive.

7.11 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. Note that none of these requirements apply to devices which only use the 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 Service for a Rights Issuer with which it is registered, the device SHALL listen to that service.

8 Service Subscription and Purchase

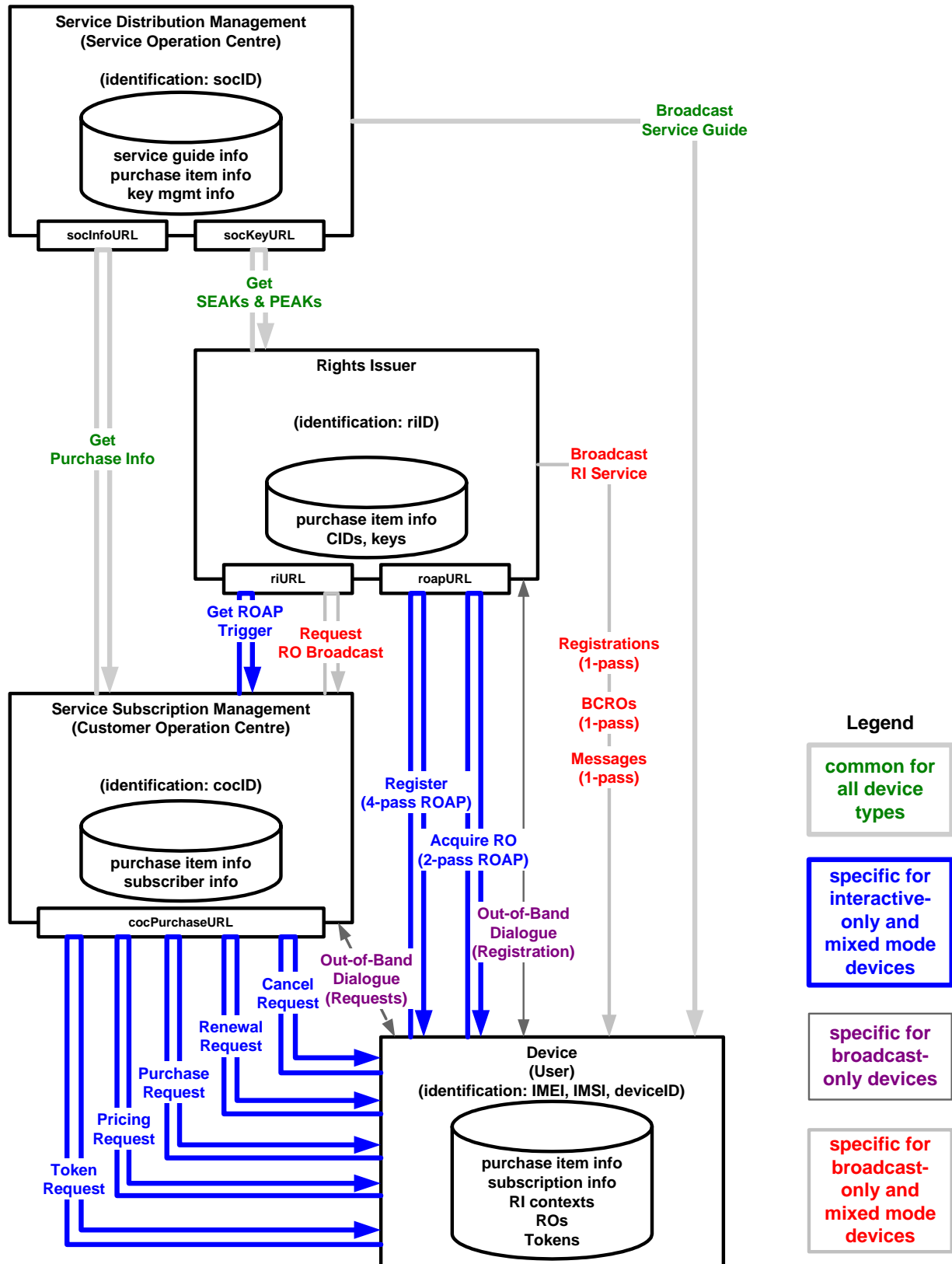


Figure 40: Message Flows for Service Subscription and Purchase

8.1 Purchase over the Interactivity Channel

Interactive devices MAY use the interactivity channel for purchase transactions. The protocol and messages of these purchase transactions are specified later in this section.

A device supporting purchase over the interactivity channel SHALL implement the protocol for communication with the network elements (SOC, COC, RI) defined in this section. However, the messages exchanged between network elements are listed here only as an informative reference.

Messages specified in this section apply also to mixed-mode devices. However for mixed-mode devices the RI MAY choose to deliver a BCRO instead of an ICRO (see section 8.2).

8.1.1 Typical Purchase Sequences

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 architecture 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 document. If it is a device supporting mixed-mode registration, it is also assumed to be able to receive BCROs via the broadcast channel.

8.1.1.1 Bulk Download of Service and Program Keys

In cases where the RI is part of the SOC, the download of service and program keys from the key generator within Service Distribution Management to the RI may be implemented as a periodical "bulk download".

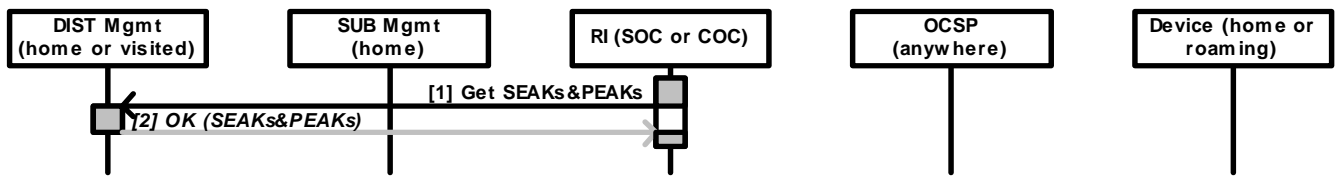


Figure 41: Interactions for Bulk Download of Service and Program Keys

1	<p>Get SEAKs & PEAKs</p> <p>The SEAKs and PEAKs are downloaded from the Service Distribution Management to the RI, together with the purchase item identification.</p> <p>SEAKs and PEAKs have a time span of validity (indicating during which time interval they are effectively used to protect broadcast traffic).</p> <p>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.</p> <p>With each purchase item, there may be some information regarding usage rules specified, which influence the generation of the rights expression by the RI.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> time interval for which associations of purchase items and their respective SEAKs and PEAKs should be returned <p><i>This message is network-internal, and is not further specified.</i></p>
2	<p>OK (SEAKs & PEAKs)</p> <p>The answer to a 'Get SEAKs & PEAKs' request contains a set of associations between purchase item identification and the corresponding SEAKs or PEAKs.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> list of purchase item associations, containing: <ul style="list-style-type: none"> purchase item ID time span of validity of contained SEAKs and PEAKs list of key records, containing <ul style="list-style-type: none"> CID SEAK or PEAK usage rule info <p><i>This message is network-internal, and is not further specified.</i></p>

8.1.1.2 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”.

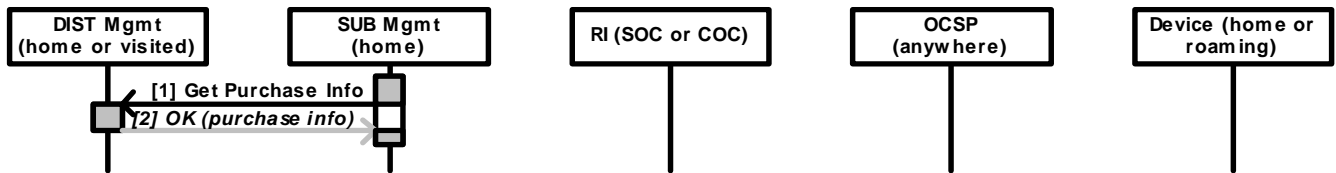


Figure 42: Interactions for Bulk Download of Purchase Information

1	<p>Get Purchase Info</p> <p>Information about purchase items (e.g. 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.</p> <p>In the case that items that can be subscribed to (service bundles), the subscription options may be indicated to the Service Subscription Management.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> time interval for which associations of purchase items and their respective composition and purchase options should be returned <p><i>This message is network-internal, and is not further specified.</i></p>
---	---

2 **OK (purchase info)**

The answer to the 'Get Purchase Info' request contains the purchase info of 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 (incl. 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.

8.1.1.3 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.

Should not all purchase items be 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 that a particular purchase item may be obtained via multiple purchase options (e.g. 1 month subscription, 12 months 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 inquire 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

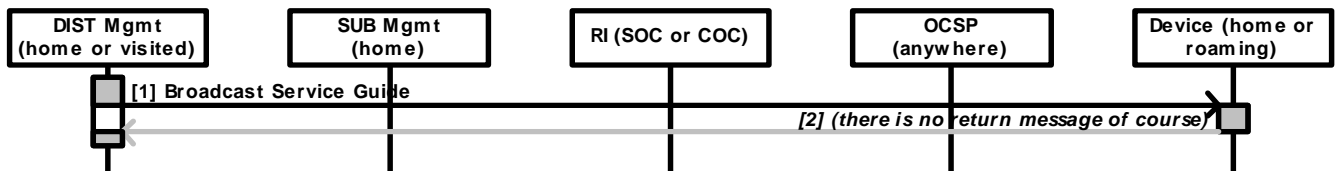


Figure 43: Interactions for Announcement of Purchase Items in Service Guide

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 the broadcast channel is assumed to be the normal case, the service guide may also be retrievable over the interactivity channel). <i>The service guide is not in scope of this specification, however, some purchase-relevant attributes are specified and are assumed to be included in the service guide.</i>
2	(there is no return message of course) The return message shown in the interaction diagram is an artefact of the diagram generator. The service guide is assumed to be distributed over broadcast, without need for interaction.

8.1.1.4 Pricing Inquiry

In the case that 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) the service guide 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.

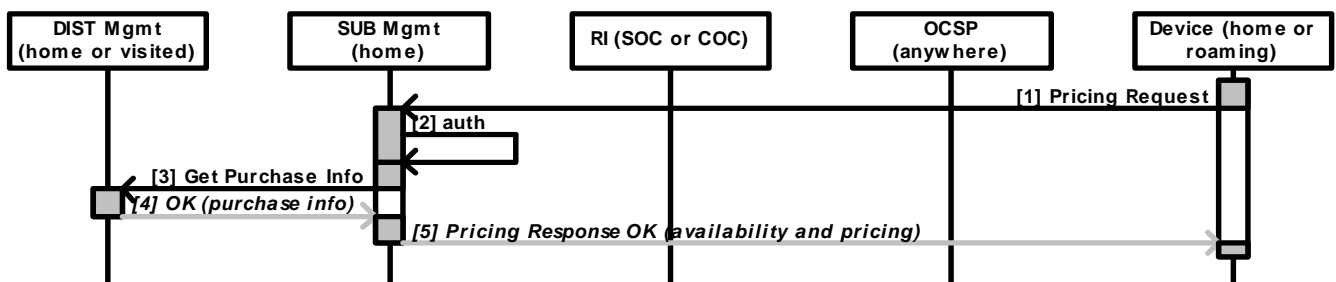


Figure 44: Interactions for Pricing Inquiry

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. <i>This message is further specified in this chapter.</i>
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. <i>This operation is network-internal, and is not further specified.</i>
3	Get Purchase Info In case the COC doesn't do "bulk download" of purchase and pricing information from the SOC, and very typically in the case that the COC has a different geographical location (e.g. different country) than the SOC, the COC inquires availability and pricing information from the SOC specifically for the requested purchase item. Recommended content of the message: <ul style="list-style-type: none"> • purchase item ID • preferred language <i>This message is network-internal, and is not further specified.</i>

4	<p>OK (purchase info)</p> <p>In the case that the purchase item requested by the device is valid, the SOC returns its purchase and pricing information (e.g. there may be multiple purchasing options with different pricing) back to the COC.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • 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) <ul style="list-style-type: none"> - 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 <ul style="list-style-type: none"> - purchase option ID - purchase option description (user-specified or default language) - purchase option price (incl. currency) <p>The list of purchase item records is not strictly necessary for COC to obtain. However, it can be assumed that for purposes of customer care it will be important for the COC to know what items are included in a purchase item.</p> <p><i>This message is network-internal, and is not further specified.</i></p>
5	<p>Pricing Response (availability and pricing)</p> <p>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.</p> <p>The successful pricing response lists all pricing options and their prices.</p> <p>The text within the pricing response may be in the user's preferred language, if supported by SOC, otherwise in a default language.</p> <p><i>This message is further specified in this chapter.</i></p>

8.1.1.5 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 an Service Subscription Management is not in scope of this specification.

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.

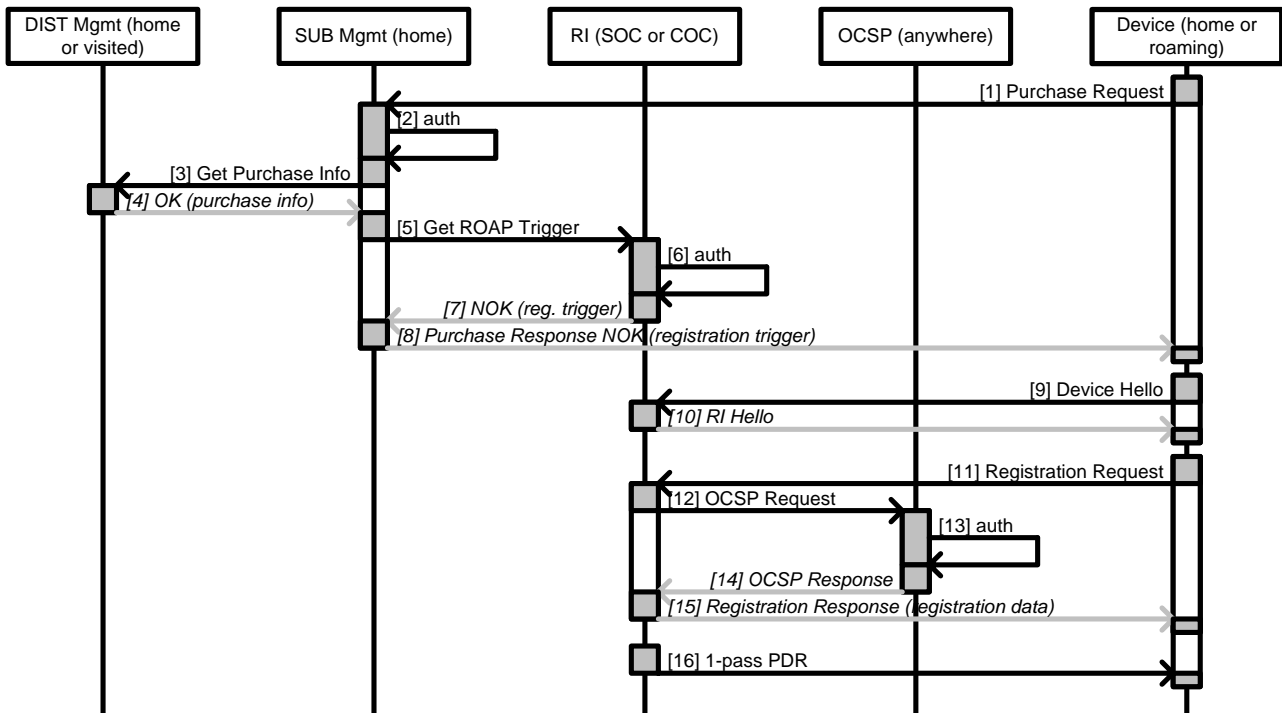


Figure 45: Interactions for Unsuccessful Purchase

1	<p>Purchase Request</p> <p>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 multiple of them).</p> <p>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.</p> <p><i>This message is further specified in this chapter.</i></p>
2	<p>authenticate</p> <p>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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>

3	<p>Get Purchase Info (optional step)</p> <p>In the case that the Service Subscription Management doesn't do "bulk download" of purchase and pricing information from the Service Distribution Management, and very typically in the case that the Service Subscription Management has a different geographical location (e.g. different country) than the Service Distribution Management, the Service Subscription Management inquires availability and pricing information from the Service Distribution Management specifically for the requested purchase item.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • purchase item ID • preferred language <p><i>This operation is network-internal, and is not further specified.</i></p>
4	<p>OK (purchase info)</p> <p>In the case that the purchase item requested by the device is valid, the Service Distribution Management returns its purchase and pricing information (e.g. there may be multiple purchasing options with different pricing) back to the Service Subscription Management.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • 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) <ul style="list-style-type: none"> - 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 <ul style="list-style-type: none"> - purchase option ID - purchase option description (in user-specified or default language) - purchase option price (incl. currency) <p>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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>

5	<p>Get ROAP Trigger</p> <p>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 (e.g. this can be the own RI of the Service Subscription Management, or the RI of the Service Distribution Management).</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • RI device ID • RI domain ID (optionally used in case the device requests the RO to be valid for a local 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) <p><i>This operation is network-internal, and is not further specified.</i></p>
6	<p>authenticate</p> <p>Before doing any further processing, the RI authenticates the device.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
7	<p>NOK (reg. trigger)</p> <p>In case the device is found to have no valid registration (e.g. 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.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • trigger for device registration (the trigger includes the rights issuer URL to which the registration can be initiated) <p><i>This operation is network-internal, and is not further specified.</i></p>
8	<p>Purchase Response NOK (registration trigger)</p> <p>The negative response from the RI, including the registration trigger, is forwarded to the device. The ROAP trigger includes the URL of the RI that the device is supposed to register with.</p> <p><i>This message is further specified in this chapter.</i></p>
9	<p>Device Hello</p> <p>Using the information contained in the trigger, the device initiates the 4-pass device registration.</p> <p><i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i></p>
10	<p>RI Hello</p> <p><i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i></p>

11	Registration Request <i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i>
12	OCSP Request <i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i>
13	authenticate Before doing any further processing, the OCSP authenticates the device. <i>This operation is network-internal, and is not further specified.</i>
14	OCSP Response <i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i>
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. <i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i>
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 section 6.4.3.4).

8.1.1.6 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.

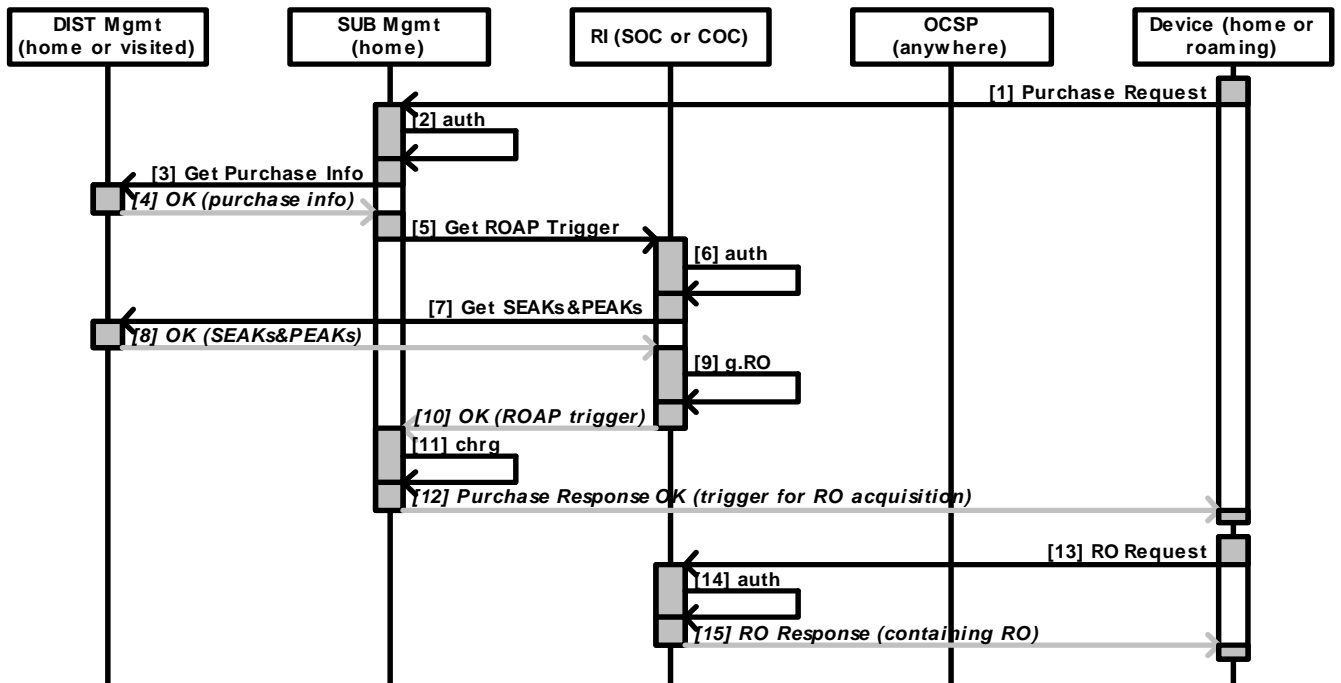


Figure 46: Interactions for Successful Purchase

1	<p>Purchase Request</p> <p>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 multiple of them).</p> <p>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.</p> <p><i>This message is further specified in this chapter.</i></p>
2	<p>authenticate</p> <p>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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>

3	<p>Get Purchase Info (optional step)</p> <p>In the case that the Service Subscription Management doesn't do "bulk download" of purchase and pricing information from the Service Distribution Management, and very typically in the case that the Service Subscription Management has a different geographical location (e.g. different country) than the Service Distribution Management, the Service Subscription Management inquires availability and pricing information from the Service Distribution Management specifically for the requested purchase item.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • purchase item ID • preferred language <p><i>This operation is network-internal, and is not further specified.</i></p>
4	<p>OK (purchase info)</p> <p>In case the purchase item requested by the device is valid, the Service Distribution Management returns its purchase and pricing information (e.g. there may be multiple purchasing options with different pricing) back to the Service Subscription Management.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • 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) <ul style="list-style-type: none"> - 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 <ul style="list-style-type: none"> - purchase option ID - purchase option description (in user-specified or default language) - purchase option price (incl. currency) <p>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.</p> <p><i>This operation is network-internal, and is not further specified..</i></p>

5	<p>Get ROAP Trigger</p> <p>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 (e.g. this can be the own RI of the Service Subscription Management, or the RI of the Service Distribution Management).</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • RI device ID • RI domain ID (optionally used in case the device requests the RO to be valid for a local 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) <p><i>This operation is network-internal, and is not further specified.</i></p>
6	<p>Authenticate</p> <p>Before doing any further processing, the RI authenticates the device.</p> <p>In this example of a purchase sequence, the device is found to have a valid registration (RI context).</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
7	<p>Get SEAKs & PEAKs (optional step)</p> <p>In the case that there is no “bulk download” of associations between purchase items and the corresponding keys (service keys, program 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.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • purchase item ID • current/next flag set to “current” (indicated that the “current” keys are requested) <p><i>This operation is network-internal, and is not further specified.</i></p>

8	<p>OK (SEAKs & PEAKs)</p> <p>The service and program keys corresponding to the purchase item as specified in the request are returned by the Service Distribution Management to the RI.</p> <p>There may be some information regarding usage rules specified, which influence the generation of the rights expression by the RI.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • purchase item ID • time span of validity of contained SEAKs or PEAK • list of key records, containing <ul style="list-style-type: none"> - CID - SEAK or PEAK • usage rule info <p><i>This operation is network-internal, and is not further specified.</i></p>
9	<p>generate RO</p> <p>The RI generates the RO containing the desired keys (e.g. multiple SEAKs in case of a subscription to a service bundle, or a PEAK in case of pay-per-view of a service).</p> <p>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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
10	<p>OK (ROAP trigger)</p> <p>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.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • 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 section 8.2). <p>The device certificate may also be returned as a result of this operation.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
11	<p>Charge</p> <p>Before sending the RO acquisition trigger back to the device, the Service Subscription Management may charge the user (if consumption-based charging based on metering is used, the user is not charged when the “purchase request” is made, but when tokens are requested).</p> <p>How charging is done (e.g. 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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>

12	<p>Purchase Response OK (trigger for RO acquisition)</p> <p>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.</p> <p>Alternatively, if the device is a mixed-mode device the RI MAY decide to deliver a BCRO over the 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 section 8.2).</p> <p>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.</p> <p><i>This message is further specified in this chapter.</i></p>
13	<p>RO Request</p> <p>Using the information contained in the ROAP trigger, the device initiates the 2-pass ROAP.</p> <p><i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i></p>
14	<p>authenticate</p> <p>Before doing any further processing, the RI authenticates the device and/or user.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
15	<p>RO Response</p> <p>As a result of a successful RO acquisition, the RO is available in the device.</p> <p><i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i></p>

8.1.1.7 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 “datetime” 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 “datetime” restriction in the “access” permission in the RO. The device SHALL request the next RO at a random point in time T, where $T1 \leq T \leq T2 - DT$.

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.

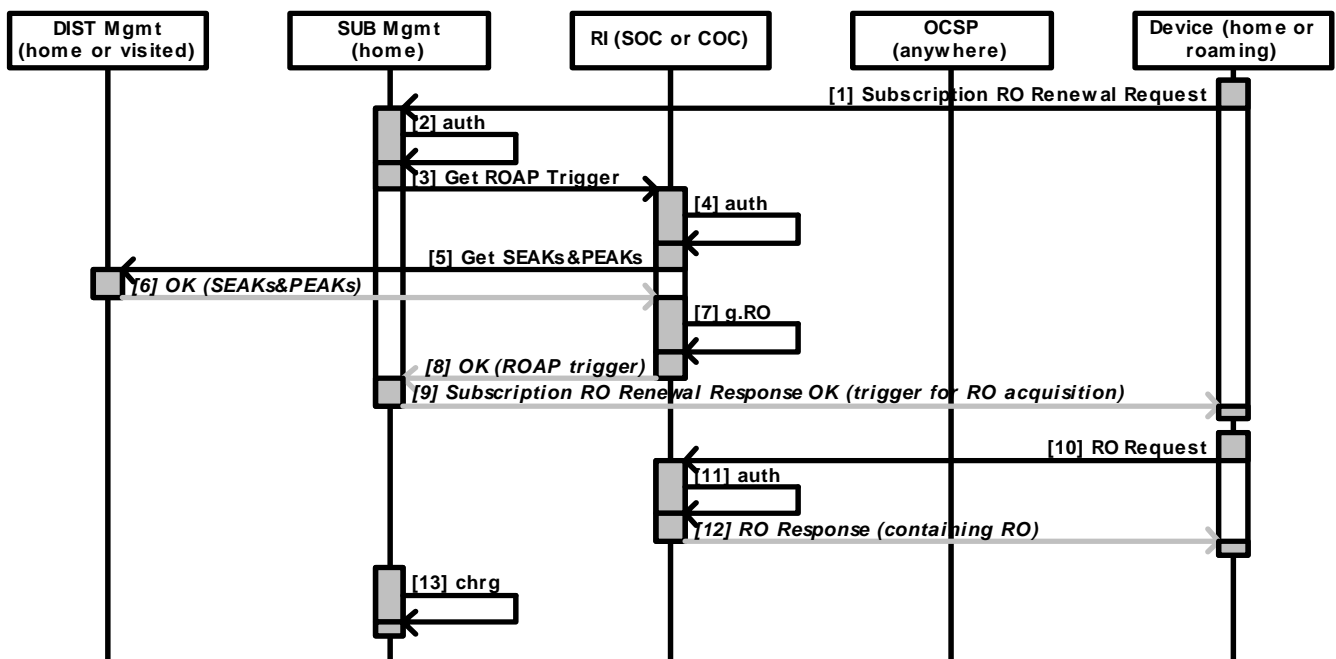


Figure 47: Interactions for Subscription RO Renewal and Asynchronous Charging

1	<p>Subscription RO Renewal Request</p> <p>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.</p> <p>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.</p> <p><i>This message is further specified in this chapter.</i></p>
2	<p>authenticate</p> <p>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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
3	<p>Get ROAP Trigger</p> <p>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.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • RI device ID • RI domain ID (optionally used in case the device requests the RO to be valid for a local 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) <p><i>This operation is network-internal, and is not further specified</i></p>
4	<p>authenticate</p> <p>Before doing any further processing, the RI authenticates the device.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>

5	<p>Get SEAKs & PEAKs (optional step)</p> <p>In the case that there is no “bulk download” of associations between purchase items and the corresponding keys (service encryption & authentication keys, program encryption & authentication keys) from the Service Distribution Management to the RI, the RI requests the necessary service encryption & authentication keys by sending the purchase item identification to the Service Distribution Management.</p> <p>Importantly, the Service Subscription Management will specify whether the “current” or “next” keys are desired.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • purchase item ID • current/next flag set to “current” (indicated that the “current” keys are requested) <p><i>This operation is network-internal, and is not further specified.</i></p>
6	<p>OK (SEAKs & PEAKs)</p> <p>The SEAKs or PEAKs corresponding to the purchase item as specified in the request are returned by the Service Distribution Management to the RI.</p> <p>There may be some information regarding usage rules specified, which influence the generation of the rights expression by the RI.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • purchase item ID • time span of validity of contained SEAKs or PEAK • list of key records, containing <ul style="list-style-type: none"> - CID - SEAK or PEAK • usage rule info <p><i>This operation is network-internal, and is not further specified.</i></p>
7	<p>generate RO</p> <p>The RI generates the RO containing the desired keys.</p> <p>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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
8	<p>OK (ROAP trigger)</p> <p>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.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • trigger for the rights object acquisition (the trigger includes the rights issuer URL from which the RO can be acquired by the device) <p><i>This operation is network-internal, and is not further specified.</i></p>

9	<p>Subscription RO Renewal Response OK (trigger for RO acquisition)</p> <p>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.</p> <p>Alternatively, if the device is a mixed-mode device the RI MAY decide to deliver a BCRO over the 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 section 8.2).</p> <p><i>This message is further specified in this chapter.</i></p>
10	<p>RO Request</p> <p>Using the information contained in the ROAP trigger, the device initiates the 2-pass ROAP.</p> <p><i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i></p>
11	<p>Authenticate</p> <p>Before doing any further processing, the RI authenticates the device and/or user.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
12	<p>RO Response</p> <p>As a result of a successful RO acquisition, the RO is available in the device.</p> <p><i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i></p>
13	<p>Charge</p> <p>How charging is done (e.g. 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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>

8.1.1.8 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.

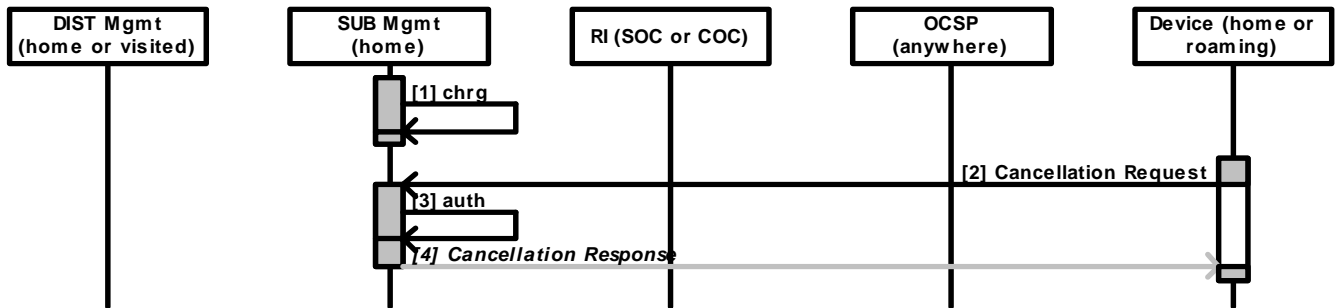


Figure 48: Interactions for Asynchronous Charging and Cancellation of Open-Ended Subscriptions

1	<p>charge</p> <p>How charging is done (e.g. 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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
2	<p>Cancellation Request</p> <p>The device sends the cancellation request to the same Service Subscription Management from where it originally purchased the subscription.</p> <p>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.</p> <p>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 fail thereafter.</p> <p><i>This message is further specified in this chapter.</i></p>
3	<p>authenticate</p> <p>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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
4	<p>Cancellation Response</p> <p>The cancellation response includes a text which 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.</p> <p><i>This message is further specified in this chapter.</i></p>

8.1.1.9 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 metering-based pre- and post-paid models.

The purchase of tokens is fully separated from the consumption metering, and independent of the existence of particular purchase items. However, it may depend on the contract a user has with the SUB Mgmt 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 are specified in [OMA-DRM-DRM]. [Note to the editor: the token features are assumed to be included in the final version of the OMA DRM 2.0 specification.]

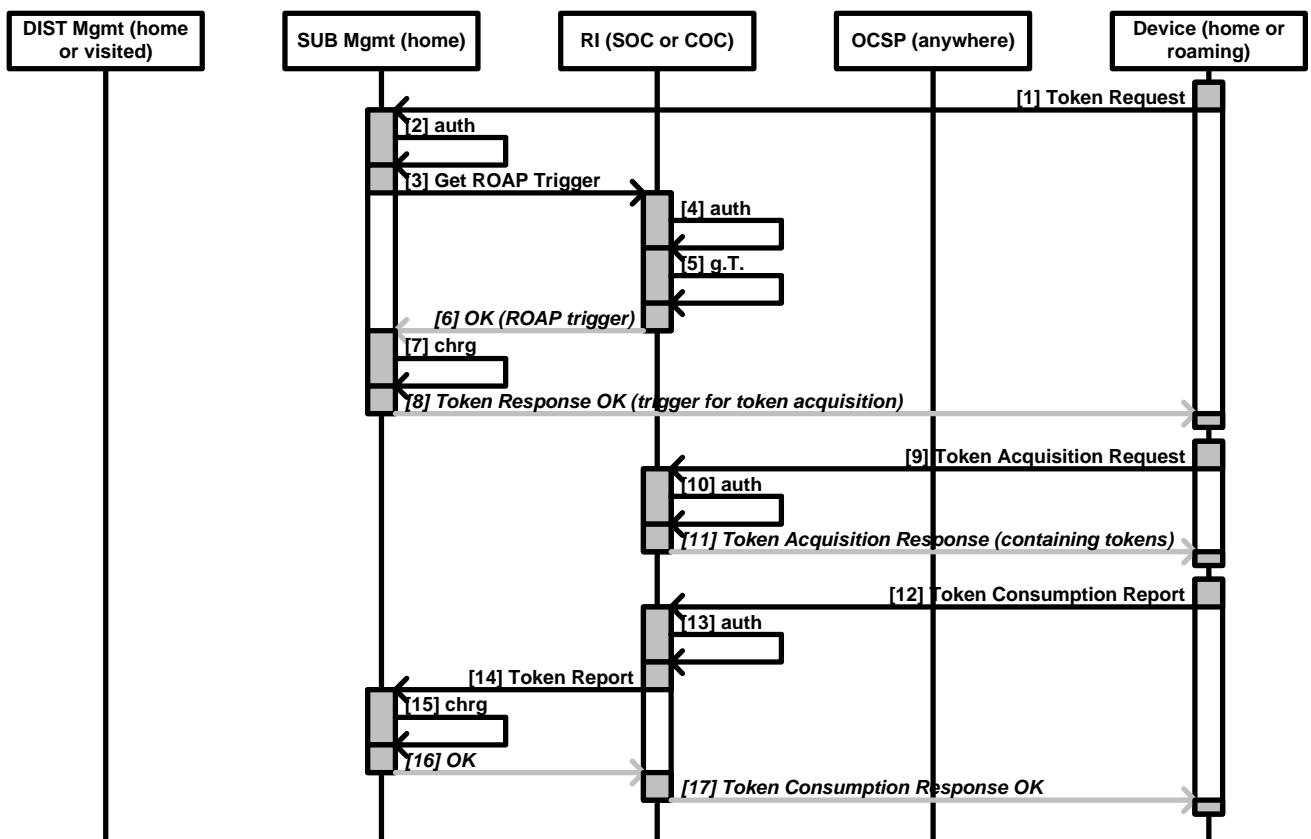


Figure 49: Interactions for Acquisition and Charging of Tokens

1	<p>Token Request</p> <p>Based on an internal logic, which is not in the scope of this specification, the terminal or user decides to purchase additional tokens from the SUB Mgmt, and issues a token request.</p> <p>The attributes of the token request are:</p> <ul style="list-style-type: none"> • 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). <p><i>This message is further specified in this chapter.</i></p>
2	<p>authenticate</p> <p>Before doing any further processing, the SUB Mgmt 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.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
3	<p>Get ROAP Trigger</p> <p>The SUB Mgmt 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 SUB Mgmt will use the same RI as for the original purchase request.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • RI device ID • number of tokens • in the case of post-paid, an indication that metering is requested for this token delivery and the URL to which the RI should send the metering report. <p><i>This operation is network-internal, and is not further specified</i></p>
4	<p>Authenticate</p> <p>Before doing any further processing, the RI authenticates the terminal and/or user.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
5	<p>generate Token</p> <p>The RI generates the desired amount of tokens.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>
6	<p>OK (ROAP trigger)</p> <p>In the case of success, the RI sends an OK message back to the SUB Mgmt, containing also the ROAP trigger that the terminal may use to request the prepared tokens.</p> <p>Recommended content of the message:</p> <ul style="list-style-type: none"> • trigger for the rights object acquisition (the trigger includes the rights issuer URL from which the token RO can be acquired by the terminal) <p>The device certificate may also be returned as a result of this operation.</p> <p><i>This operation is network-internal, and is not further specified.</i></p>

7	Charge (pre-paid only) How charging is done (e.g. 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 SUB Mgmt. This step is executed only if pre-paid charging is selected. <i>This operation is network-internal, and is not further specified.</i>
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. <i>This message is further specified in this chapter.</i>
9	Token Acquisition Request Using the information contained in the ROAP trigger, the terminal initiates the 2-pass ROAP. <i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i>
10	Authenticate Before doing any further processing, the RI authenticates the terminal and/or user. <i>This operation is network-internal, and is not further specified.</i>
11	Token Acquisition Response As a result of a successful token acquisition, the token is available in the terminal. <i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i>
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. <i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i>
13	Authenticate Before doing any further processing, the RI authenticates the terminal and/or user. <i>This operation is network-internal, and is not further specified.</i>
14	Token Report After having validated it, the RI forwards the token report to the SUB Mgmt. <i>This operation is network-internal, and is not further specified.</i>
15	Charge The SUB Mgmt charges the user based on the amount of tokens consumed. <i>This operation is network-internal, and is not further specified.</i>
16	OK <i>This operation is network-internal, and is not further specified.</i>
17	Token Consumption Report OK <i>This is a standard OMA DRM 2.0 operation, and is not further specified.</i>

8.1.2 Protocol

The Service Provider SHALL support HTTP POST, which MAY be used for purchase requests over the Interactivity channel.

The Service Provider MAY support HTTPS POST, which MAY be used for purchase requests over the Interactivity channel.

The Service Provider SHALL use either HTTP POST or HTTPS POST for purchase requests over the Interactivity channel.

The Device SHALL support both HTTP POST and HTTPS POST for purchase requests over the 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.

8.1.2.1 HTTP Headers

Request messages are sent as HTTP content of type “application/xml”. Responses are always sent as part of the “200 OK” response to the original request. The content type is “application/xml” if the response has only one payload, or “multipart/mixed” in the case of multiple payloads (e.g. if the response includes one or more ROAP triggers). In the latter case, the content type of a single payload of the multipart content will be “application/xml” for the messages defined here, and “application/vnd.oma.drm.roap-trigger+xml” for the ROAP triggers.

8.1.2.2 Signatures

All request messages SHALL be signed with the Private Device Key, using the RSASSA-PSS algorithm [PKCS#1] (see also A.11). The input to the signature operation SHALL be the XML payload of the request without the “signature” element, canonicalized according to the Exclusive XML Canonicalization algorithm [XML_XCAN].

8.1.3 XML Schemas for Request and Response Messages

8.1.3.1 Basic Types

This section contains the XML schema fragment definitions for the elementary data types used in the subsequent message definitions.

8.1.3.1.1 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 [ISO639]). Below is the XML schema fragment for the UserData Type:

```
<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>
```

8.1.3.1.2 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 deviceID 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-DRM-DRM].
- the broadcast bearers supported by the device (currently defined identifiers: “dvb-h”, “mbms”, “bcmcs”).
- the service protection protocols supported by the device (currently defined identifiers: “ipsec”, “srtp”).
- 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:string"/>
    <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>
```

8.1.3.1.3 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 a OMA DRM 2.0 domain (“omadrm2”) or a Local 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>
```

8.1.3.1.4 ServiceOperationCentreType

The Service Operation Centre data contains:

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 chapter 10.2). 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 the table below:

Table 70: 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					

Legend: “M” means that the element is MANDATORY for the particular request.

Below is the XML schema fragment for the ServiceOperationCentreType:

```
<xs:complexType name="serviceOperationCentreType">
  <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>
```

8.1.3.1.5 PriceType

The price information contains the price as a decimal value and an optional currency qualifier (specified as defined in [ISO4217]), 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>
```

8.1.3.1.6 Purchase Item Type

The purchase item data contains the information needed to identify the user’s purchase:

ItemID: the purchaseItemID received from the service guide.

PurchaseOption: an identifier of the particular type of purchase, e.g. open-ended subscription or one-time subscription, a received from the service guide.

Price: the price of the item known to the user (optional)

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="purchaseOption" type="xs:token" minOccurs="0"/>
    <xs:element name="price" type="priceType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

8.1.3.1.7 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>
```

8.1.3.1.8 Response Type

Each response message extends the base response type. The response type has a “status” attribute, which is either success or error, and a “reason code” attribute that indicates the cause of error among the ones listed in table 3.

```
<xs:complexType name="responseType">
  <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>
```

8.1.3.2 Error Codes

The following table lists all the possible reasonCode values for error case, and their applicability to each transaction.

Table 71: Occurrence of Error Codes in Response Messages

Code	Error Situation	Pricing	Purchase	Renewal	Cancel	Token
0	<p>Authentication Failed</p> <p>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.</p> <p>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.</p>	X	X	X	X	X
1	<p>Purchase Item Unknown</p> <p>This code indicates that the requested purchase item is unknown. This can happen e.g. if the device has a cached service guide with old information.</p> <p>In this case, the user may re-acquire the service guide.</p>	X	X			
2	<p>Device Not Authorized</p> <p>This code indicates that the device is not authorized to get ROs from the RI, e.g. because the device certificate was revoked.</p> <p>In this case, the user may contact the Service Subscription Management operator.</p>		X	X		X
3	<p>Device Not Registered</p> <p>This code indicates that the device is not registered with the RI that is used for the transaction.</p> <p>When this code is sent, the response message includes a registration trigger that allows the device to register.</p> <p>In this case, the device MAY automatically perform the registration, and, if the registration is successful, re-initiate the original transaction.</p>		X	X		X
4	<p>Server Error</p> <p>This code indicates that there was a server error, such as a problem connecting to a remote back-end system.</p> <p>In such a case, the transaction may succeed if it is re-initiated later.</p>	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 (e.g. if there is an interoperability problem) succeed if it is re-initiated later.	X	X	X	X	X
6	Charging Error This code indicates that the charging step failed (e.g. 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 it does not belong to. 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		
----	---	--	---	---	--	--

Legend: “X” means that only the particular request may result in the corresponding error code.

8.1.3.3 Pricing Request

The pricing request includes the user, device and SOC data described above, and a list of the identifiers of the purchase items for which the user is requesting the price details.

8.1.3.3.1 XML Schema

```

<xs:element name="pricingRequest" type="pricingRequestType"/>
<xs:complexType name="pricingRequestType">
  <xs:complexContent>
    <xs:extension base="requestType">
      <xs:sequence>
        <xs:element name="user" type="userDataType"/>
        <xs:element name="device" type="deviceDataType"/>
        <xs:element name="serviceOperatorCentre" type="serviceOperatorCentreType"/>
        <xs:element name="purchaseItemList">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="purchaseItem" type="purchaseItemType" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="signature" type="xs:base64Binary"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

8.1.3.3.2 Example

```

<?xml version="1.0" encoding="UTF-8"?>
<pricingRequest interfaceVersion="1">
  <user>
    <userID>24403123456</userID>
    <lang>en</lang>
  </user>
  <device>
    <deviceID>0044005817853</deviceID>
    <riDeviceID>1234567890</riDeviceID>
    <bcastBearer>dvb</bcastBearer>
    <serviceProtection>ipsec</serviceProtection>
    <deviceType>interactive</deviceType>
  </device>
  <serviceOperationCentre>
    <socID>12345</socID>
    <socInfoURL>http://www.soc.com/info</socInfoURL>
  </serviceOperationCentre>
  <purchaseItemList>
    <purchaseItem>
      <itemID>5432</itemID>
    </purchaseItem>
  </purchaseItemList>
  <signature>f7jwx3rvEP00vKtMup4NbeVu9kn=</signature>

```

```
</pricingRequest>
```

8.1.3.4 Pricing Response

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.

8.1.3.4.1 XML Schema

```
<xs:element name="pricingResponse" type="pricingResponseType"/>
<xs:complexType name="pricingResponseType">
  <xs:complexContent>
    <xs:extension base="responseType">
      <xs:sequence minOccurs="0">
        <xs:element name="datacastOperator" type="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="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:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

8.1.3.4.2 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">
```



```

<serviceOperationCentre>
  <socID>12345</socID>
</serviceOperationCentre>
<purchaseItemList>
  <purchaseItem itemID="5432">
    <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>

```

8.1.3.5 Purchase Request

The purchase request includes the user, device and SOC data described 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.

8.1.3.5.1 Schema

```

<xs:element name="purchaseRequest" type="purchaseRequestType"/>
<xs:complexType name="purchaseRequestType">
  <xs:complexContent>
    <xs:extension base="requestType">
      <xs:sequence>
        <xs:element name="user" type="userDataType"/>
        <xs:element name="device" type="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="serviceOperatorCentreType"/>
        <xs:element name="purchaseItemList">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="purchaseItem" type="purchaseItemType" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="signature" type="xs:base64Binary"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

8.1.3.5.2 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>

```

```

</device>
<domain>
  <domainID>dhf5434jddAdfD</domainID>
</domain>
<serviceOperationCentre>
  <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>
</serviceOperationCentre>
<purchaseItemList>
  <purchaseItem>
    <itemID>5432</itemID>
    <purchaseOption>B</purchaseOption>
    <price currency="EUR">5.00</price>
  </purchaseItem>
</purchaseItemList>
<signature>h8twx3rvEP00vKtMup4NbeVu0f1O</signature>
</purchaseRequest>

```

8.1.3.6 Purchase Response

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.

8.1.3.6.1 XML Schema

```

<xs:element name="purchaseResponse" type="purchaseResponseType"/>
<xs:complexType name="purchaseResponseType">
  <xs:complexContent>
    <xs:extension base="responseType">
      <xs:sequence>
        <xs:element name="serviceOperatorCentre" type="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>

```

8.1.3.6.2 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"/>

```

8.1.3.6.3 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"/>
```

8.1.3.6.4 Example: Unsuccessful Purchase Response with Purchase-Item-specific Error

```
<?xml version="1.0" encoding="UTF-8"?>
<purchaseResponse status="error">
  <serviceOperationCentre>
    <socID>12345</socID>
  </serviceOperationCentre>
  <purchaseItemList>
    <purchaseItem itemID="5432">
      <reasonCode>1</reasonCode>
    </purchaseItem>
  </purchaseItemList>
</purchaseResponse>
```

8.1.3.7 Subscription RO Renewal Request

The subscription renewal request includes the user, device and SOC data described above, and a list of the identifiers of purchase items corresponding to subscriptions the user wants to renew.

8.1.3.7.1 XML Schema

```
<xs:element name="renewalRequest" type="renewalRequestType"/>
<xs:complexType name="renewalRequestType">
  <xs:complexContent>
    <xs:extension base="requestType">
      <xs:sequence>
        <xs:element name="user" type="userDataType"/>
        <xs:element name="device" type="deviceDataType"/>
        <xs:element name="serviceOperatorCentre" type="serviceOperatorCentreType"/>
        <xs:element name="purchaseItemList">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="purchaseItem" type="purchaseItemType" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="signature" type="xs:base64Binary"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

8.1.3.7.2 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>dvb</bcastBearer>
    <serviceProtection>ipsec</serviceProtection>
    <deviceType>interactive</deviceType>
  </device>
  <serviceOperationCentre>
    <socID>12345</socID>
    <socKeyURL>http://www.soc.com/keys</socKeyURL>
    <riID>4567</riID>
    <riURL>http://www.soc.com/ri</riURL>
  </serviceOperationCentre>
  <purchaseItemList>
    <purchaseItem>
      <itemID>5432</itemID>
```

```

    </purchaseItem>
  </purchaseItemList>
  <signature>q2ewx3rvEP00vKtMup4NbeVulwd9</signature>
</renewalRequest>

```

8.1.3.8 Subscription RO Renewal Response

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.

8.1.3.8.1 Schema

```

<xs:element name="renewalResponse" type="renewalResponseType"/>
<xs:complexType name="renewalResponseType">
  <xs:complexContent>
    <xs:extension base="responseType">
      <xs:sequence>
        <xs:element name="serviceOperatorCentre" type="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>

```

8.1.3.8.2 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>

```

8.1.3.8.3 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"/>

```

8.1.3.8.4 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">
  <serviceOperationCentre>
    <socID>12345</socID>
  </serviceOperationCentre>
  <purchaseItemList>
    <purchaseItem itemID="5432">
      <reasonCode>7</reasonCode>
    </purchaseItem>
  </purchaseItemList>
</renewalResponse>
```

8.1.3.9 Subscription Cancellation Request

The subscription cancellation request includes the user, device and SOC data described above, and a list of the identifiers of purchase items corresponding to subscriptions the user wants to cancel.

8.1.3.9.1 XML Schema

```
<xs:element name="cancellationRequest" type="cancellationRequestType"/>
<xs:complexType name="cancellationRequestType">
  <xs:complexContent>
    <xs:extension base="requestType">
      <xs:sequence>
        <xs:element name="user" type="userDataType"/>
        <xs:element name="device" type="deviceDataType"/>
        <xs:element name="serviceOperatorCentre" type="serviceOperatorCentreType"/>
        <xs:element name="purchaseItemList">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="purchaseItem" type="purchaseItemType" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="signature" type="xs:base64Binary"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

8.1.3.9.2 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>dvh</bcastBearer>
    <serviceProtection>srt</serviceProtection>
    <deviceType>interactive</deviceType>
  </device>
  <serviceOperationCentre>
    <socID>12345</socID>
  </serviceOperationCentre>
  <purchaseItemList>
    <purchaseItem>
      <itemID>5432</itemID>
    </purchaseItem>
  </purchaseItemList>
  <signature>w2jwx3rvEP00vKtMup4NbeVu9en1</signature>
</cancellationRequest>
```

8.1.3.10 Subscription Cancellation Response

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.

8.1.3.10.1 Schema

```
<xs:element name="cancellationResponse" type="cancellationResponseType" />
<xs:complexType name="cancellationResponseType">
  <xs:complexContent>
    <xs:extension base="responseType">
      <xs:sequence>
        <xs:element name="serviceOperatorCentre" type="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>
```

8.1.3.10.2 Example: Successful Cancellation Response

```
<?xml version="1.0" encoding="UTF-8"?>
<cancellationResponse status="success">
  <serviceOperationCentre>
    <socID>12345</socID>
  </serviceOperationCentre>
  <purchaseItemList>
    <purchaseItem itemID="5432">
      <cancellationInfoMessage>Service valid until 2005-03-27</cancellationInfoMessage>
    </purchaseItem>
  </purchaseItemList>
</cancellationResponse>
```

8.1.3.10.3 Example: Unsuccessful Cancellation Response With Purchase-Item-specific Error

```
<?xml version="1.0" encoding="UTF-8"?>
<cancellationResponse status="error">
  <serviceOperationCentre>
    <socID>12345</socID>
  </serviceOperationCentre>
  <purchaseItemList>
    <purchaseItem itemID="5432">
      <reasonCode>7</reasonCode>
    </purchaseItem>
  </purchaseItemList>
</cancellationResponse>
```

8.1.3.11 Token Request

In addition to the user, device and SOC data described 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 doesn't exceed the user's credit limit.

8.1.3.11.1 XML Schema

```
<xs:element name="tokenRequest" type="tokenRequestType"/>
<xs:complexType name="tokenRequestType">
  <xs:complexContent>
    <xs:extension base="requestType">
      <xs:sequence>
        <xs:element name="user" type="userDataType"/>
        <xs:element name="device" type="deviceDataType"/>
        <xs:element name="serviceOperatorCentre" type="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:negativeInteger"/>
        <xs:element name="signature" type="xs:base64Binary"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

8.1.3.11.2 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<tokenRequest interfaceVersion="1">
  <user>
    <userID>24403123456</userID>
    <lang>en</lang>
  </user>
  <device>
    <UDN>0044005817853</UDN>
    <riDeviceID>1234567890</riDeviceID>
    <bcastBearer>DVBH</bcastBearer>
    <serviceProtectionProtocol>IPsec</serviceProtectionProtocol>
    <broadcastMode>no</broadcastMode>
  </device>
  <serviceOperationCentre>
    <socID>12345</socID>
  </serviceOperatorCentre>
  <paymentType>postPaid</paymentType>
  <reportedTokens>10</reportedTokens>
  <requestedTokens>20</requestedTokens>
  <signature>d2jwx3rvEP00vKtMup4NbeVu9ke7</signature>
</tokenRequest>
```

8.1.3.12 Token Response

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).

8.1.3.12.1 Schema

```
<xs:element name="tokenResponse" type="tokenResponseType"/>
```

```
<xs:complexType name="tokenResponseType">
  <xs:complexContent>
    <xs:extension base="responseType" />
  </xs:complexContent>
</xs:complexType>
```

8.1.3.12.2 Example: Successful Token Response

```
<?xml version="1.0" encoding="UTF-8"?>
<tokenResponse status="success" />
```

8.1.3.12.3 Example: Unsuccessful Token Response

```
<?xml version="1.0" encoding="UTF-8"?>
<tokenResponse status="error" failureCode="1" />
```

8.2 Purchase for Mixed-Mode Devices

The sequences described in section 8.1.1 apply also to mixed-mode devices. However, for mixed-mode devices the RI MAY decide to deliver a BCRO over the 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 as described in section 6.3.4. The flows in section 8.1.1 are thus modified as follows:

Unsuccessful purchase (figure 43, 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 44, step 12): the device receives the “success” response but no RO acquisition trigger. The device SHALL prepare to receive a BCRO.

RO renewal (figure 45, step 9): the device receives the “success” response but no RO acquisition trigger. The device SHALL prepare to receive a BCRO.

Token Request (figure 47, 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 local domain, and domain ROs for that domain MAY be delivered via the interactivity channel. In this case, the ICRO will be addressed to the longform domain ID (identical to OMA domain ID) and the device SHALL obtain the shortform domain ID from the association between longform and shortform domain IDs it has established when it received the registration data (see section 6.4.4).

8.3 Out-of-Band Purchase

8.3.1 Means of purchase (informative)

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 (e.g. 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 section describes the normative behaviour of the device in case it is equipped with an appropriate contact() either via the ESG or via the update_contact_number_msg() message.

8.3.2 Out-of-Band purchase from service guide data (normative)

Generally, the request/response pairs for purchase over the 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 section 8.4.

In order to request a purchase, the user is required to provide the following data to the Customer Operation Centre, via the out-of-band channel indicated by the service guide:

Table 72: Data to be provided to the Customer Operation Centre

	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 (refer to A.10.2) including purchase code	ARC code (refer to A.10.2) 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. #itemName)	purchase item name (i.e. PitemName)

Note: The data in table 72 is provided by the service guide information, please refer to section 8.4 for details.

Note: A device which 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 a(n) offline browser.

An example of a displayed OOB purchase message for a registered device follows, where the following information is reported back to the RI¹³:

¹³ 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 specification. 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 MUST be included as defined above (please note: the ARC code will only be displayed after the first registration, when that data MAY be available for display).

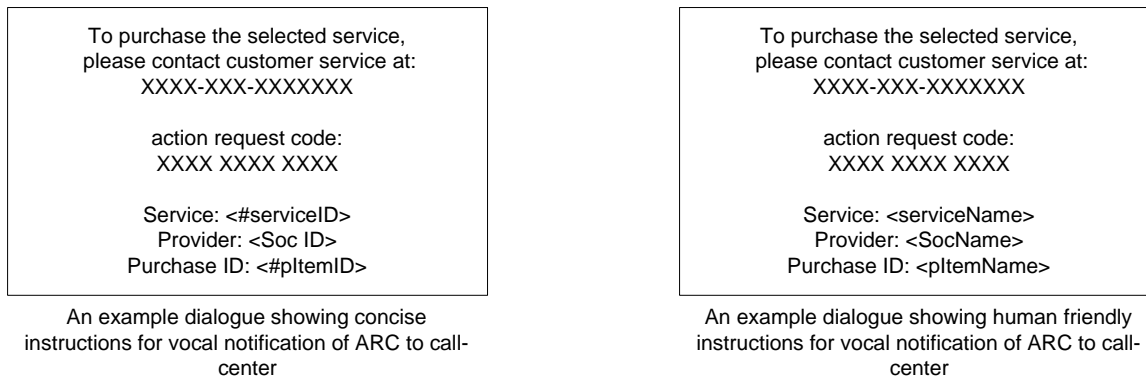


Figure 50: Samples of out-of-band purchase information displays for a registered device

Key:

- The contact can be any number as specified in contact() (Refer to section 6.4.3.7.6.2.1) or delivered per service guide.
- The action request code SHALL incorporate the purchase action request code (refer to section 6.4.3.3).

The following dialogue is an example of what an unregistered device could display.

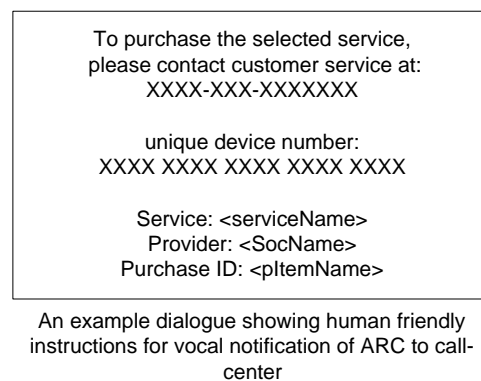


Figure 51: Sample of out-of-band purchase information displays for an unregistered device

Note: An unregistered device performing a purchase request will need to register first. Please refer to section 5.3.3 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 (c.f. chapter 7), that comprises mechanisms to broadcast BCROs spontaneously (as a result of the purchase transaction) or in a scheduled manner (e.g. for re-keying the SEAKs).

8.4 Required Service Guide Information

[Note to the editor: This chapter sets out a number of requirements on the ESG and must be checked once the ESG specification is complete and available.]

Whereas this specifications doesn't prescribe the precise structure of the service guide, the purchasing part of the service guide needs to be designed in line with the requirements derived from this chapter, 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).

Therefore, the required attributes are presented in the following figure in form of a relational diagram.

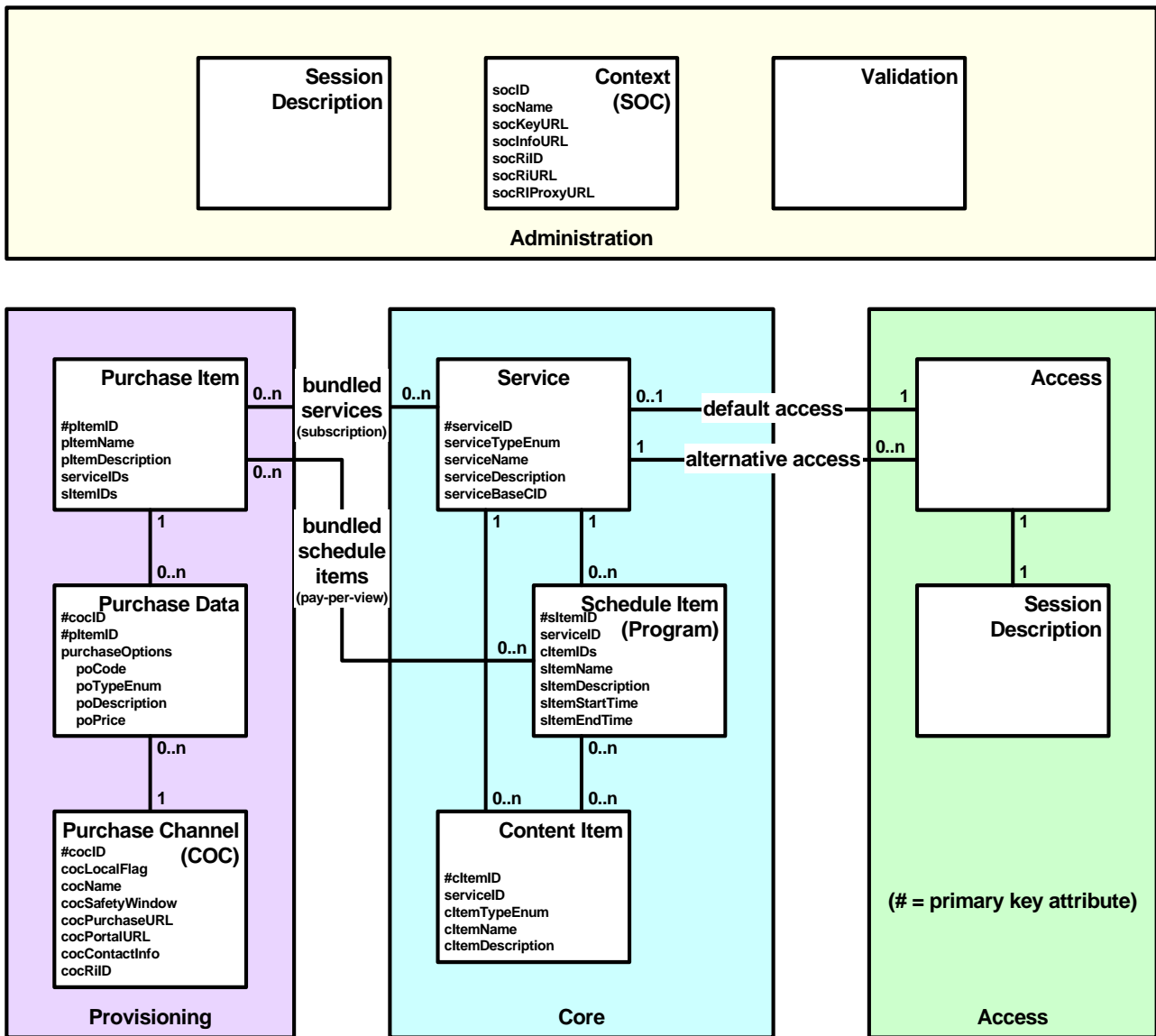


Figure 52: Service Guide Information for Service Subscription and Purchase

8.4.1 Service Operation Centre

(including Service Distribution Management):

<p>socID (mandatory, 1) – is the globally co-ordinated ID of the Service Operation Centre (aka. “socID”); the DVB platform ID MAY (and is recommended to) be used for this purpose.</p> <p>socName (mandatory, 1) – is the name of the Service Operation Centre, reasonably globally unique (e.g. 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. The name SHALL be encoded using UTF-8 [RFC3629].</p> <p>socKeyURL (mandatory, 1) – URL through which an RI can retrieve keys from Service Distribution Management.</p> <p>socInfoURL (mandatory, 1) – URL through which Service Subscription Management can retrieve purchase info from Service Distribution Management.</p> <p>socRiID (mandatory, 1) – ID of the RI associated with Service Operation Centre (used by Service Subscription Management for identifying the RI).</p> <p>socRiURL (mandatory, 1) – URL of the RI associated with Service Operation Centre (used by Service Subscription Management for requesting ROAP triggers or requesting registration data or BCROs to be broadcast)</p> <p>socRiProxyURL (optional, 1) – URL of the RI Proxy associated with Service Operation Centre (used by a Service Subscription Management in a different network to insert BCROs, as described in chapter 10.2). This parameter is present only if the RI Proxy functionality is supported by the SOC.</p>

8.4.2 Customer Operation Centre

(including Service Subscription Management):

<p>cocID (mandatory, 1) – is the globally co-ordinated ID of the Customer Operation Centre.</p> <p>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).</p> <p>cocName (mandatory, 1, multi-language) – is the name of the Customer Operation Centre, reasonably globally unique (e.g. 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 [RFC3629].</p> <p>cocRekeyingSafetyWindow (mandatory, 1) – is the time interval DT, as specified in 8.1.1.7, during which it cannot be guaranteed that RO renewal will succeed timely for uninterrupted access to a particular service.</p> <p>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 document SHALL be supported.</p> <p>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 document.</p> <p>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 (e.g. toll-free phone number, international phone number, letter address, e-mail address, SMS number, etc.).</p> <p>cocRiID (mandatory, 1) – ID of the RI associated with Customer Operation Centre (needed to allow broadcast devices to identify the RI service that may be operated by their Home COC).</p>
--

The service guide SHALL include an entry for every COC with which the SOC has an agreement (“locality agreement”, “roaming agreement”, c.f. chapter 10). 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).

8.4.3 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. The name SHALL be encoded using UTF-8 [RFC3629].

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 program keys related to the service.

8.4.4 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. The name SHALL be encoded using UTF-8 [RFC3629].

sItemDescription (optional, 0..1, multi-language) – is the description of the schedule item.

sItemStartTime – is the start time of the schedule item, specified as yyyy-mm-ddThh:mm:ss

sItemEndTime – is the end time of the schedule item, specified as yyyy-mm-ddThh:mm:ss

8.4.5 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. The name SHALL be encoded using UTF-8 [RFC3629].

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 sub-structure:

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).

8.4.6 Purchase Item

pItemID (mandatory, 1) – is the ID of the purchase item (aka. “purchaseItemID”), unique within the scope of a Service Distribution Management (socID).

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. The name SHALL be encoded using UTF-8 [RFC3629].

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 program, 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).

8.4.7 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 [OMA DRM 2, broadcast 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.

9 Head-end for Service Protection (INFORMATIVE)

The normative part of this specification defines building blocks that can be very flexibly combined into a real system deployment. A head-end architecture for service protection is proposed in this chapter, and some considerations are given that justify the architecture, but this architecture is by no means the only possible one. Therefore, this whole chapter is informative in its entirety.

9.1 Function Blocks

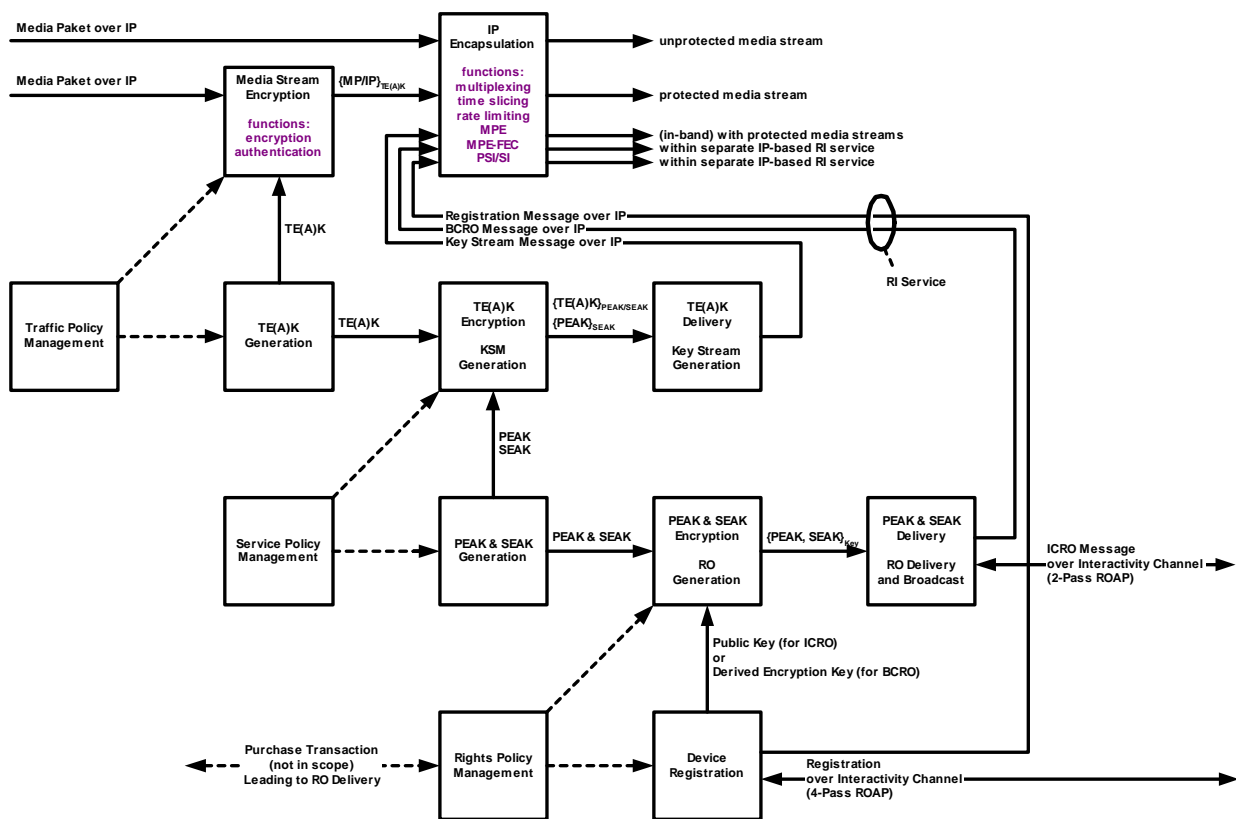


Figure 53: Function Blocks of Service Protection Head-End

The function blocks are arranged according to layers in the 4-layer model for service protection, and probably do not require further description, except for the policy management functions.

Traffic Policy Management defines which media streams need encryption, which streams are “bundled” to share their traffic key material, and the length of the crypto period.

Service Policy Management defines Services and Programmes and their lifetimes, which “bundles of media flows” (corresponding to a key stream) they contain, and the access criteria to each key stream. Specifically, this functional block also manages the PEAKs and SEAKs. In DVB terminology, this includes Event Information Scheduling.

Rights Policy Management defines purchasable items, which are “bundles of services and programmes”. It also defines to which device or domain a RO is bound and what rights the device (or its owner) has.

9.2 Considerations

The traditional security architecture of the DVB head-end, with cell-local key generation for both ECMs (corresponding to KSMs) and EMMs (corresponding to ROs), does not sufficiently cater for the needs of mobile devices, and can therefore not be used one-to-one for DVB-H.

Device mobility adds the requirement that ROs SHALL be valid for the whole network, to avoid the need for RO acquisition upon cell change. Hence, the PEAKs and SEAKs need to be generated centrally, in order for them to be packaged into ROs that have network-global validity.

TE(A)K generation can in principle still be cell-local, but this might have negative effects on cell hand-over, necessitating that, before traffic can be decrypted in the new cell, a KSM needs to be acquired and processed leading to new security associations to be set up. Local TE(A)K generation has furthermore the disadvantage that PEAKs and SEAKs need to be distributed to all encryptors, whereas with central TE(A)K generation, only the TE(A)Ks need to be distributed (optimizations are possible, such as synchronized pseudo-random generators). Therefore, in the proposal below, TE(A)K generation is shown as a central function.

The Media Stream Encryption function can very easily be separated from the IP Encapsulation. It is also possible to combine the Encryption with the Media Stream Generation function, or to separate Media Stream Encryption into its own network element. There are advantages and disadvantages with all options.

- combining encryption with the generation of Media Streams offers end-to-end protection, but may be costly to deploy, because the Encryptor has to be integrated with all possible Generators used in the network.
- combining encryption with the encapsulation of Media Streams is very easy to deploy, but then the up-stream traffic has to be protected by different means (this is not such a big down-side, since the threat model is different upstream of the Encapsulator)
- separating encryption into its own network element gives most flexibility, but then the Encryptor will have to have many functions (e.g. IP routing, multiplexing, possibly rate limiting) in common with the Encapsulator, so that both elements will turn out to be very similar

In the proposal below, the Media Stream Encryption is combined with the IP Encapsulation into a single network element, but this is by no means a requirement of the architecture.

9.3 Proposed Network Elements

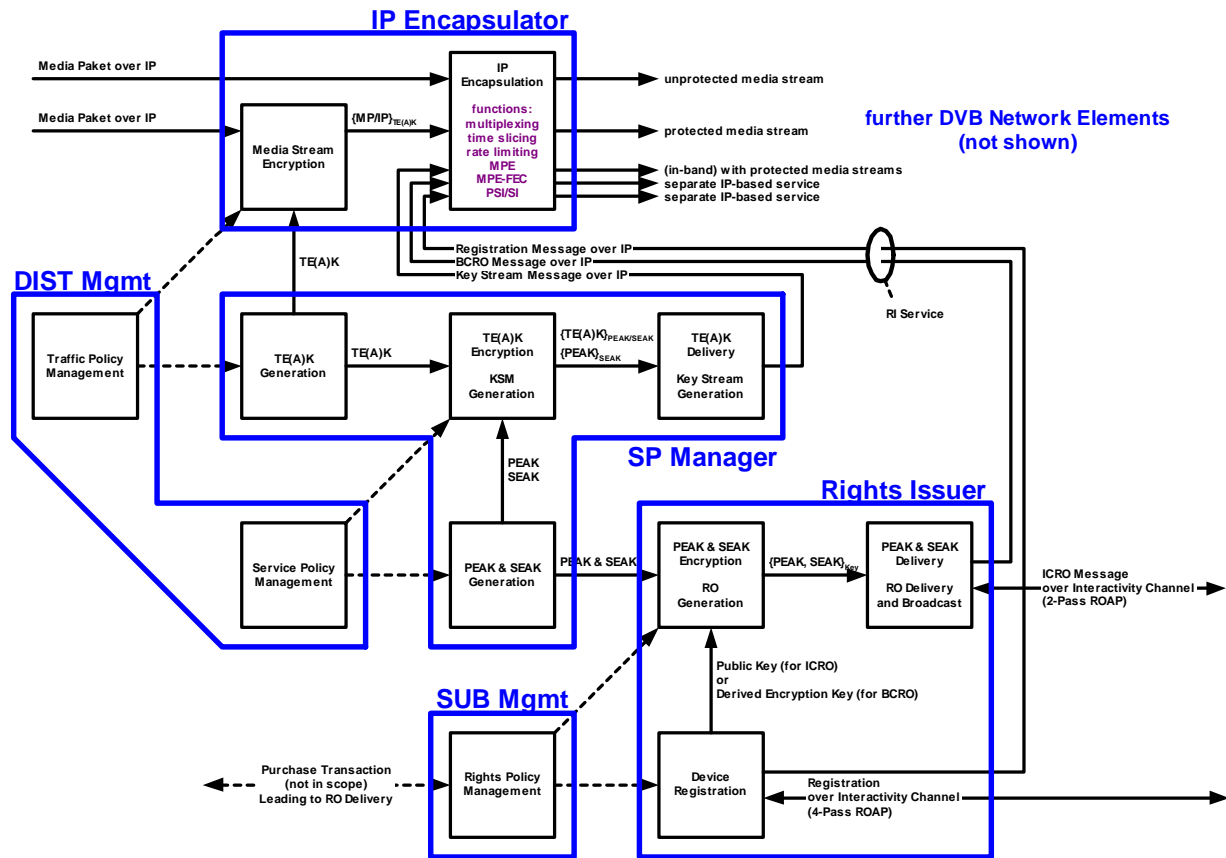


Figure 54: Systems and Network Elements of Service Protection Head-End

The proposed head-end architecture for service protection defines the following systems and network elements:

The **IP Encapsulator (IPE)** is a network element that can be deployed cell-local, but also (in case an ASI distribution network is used) network-global, or something in between. The IPE can become a “box” with well-defined standard interfaces. Upstream, the interface needs to support IP routing functionality, and the physical interfaces can be those of an IP router. Downstream, the interface needs to support MPEG-2 transport streams, and the physical interfaces can be ASI. To what extent existing DVB head-end interface specifications are re-usable for the management interfaces (especially for putting in place TE(A)Ks) needs to be further studied.

The **Service Protection Manager** and the **Rights Issuer** are network-global elements. They may be “boxes”: the functionality is well-defined, and the interfaces are minimal.

The **Distribution Management System** and the **Subscription Management System** are true systems, with differentiation based on their functionality. There are already standards for some of the interfaces, and some more of the interfaces will eventually become standardized (e.g. the interfaces needed for roaming).

10 Deployment and Roaming Scenarios (INFORMATIVE)

The deployment and roaming scenarios described in this chapter build on the purchasing mechanisms (see chapter 8) and rights issuer mechanisms (see chapter 7), and the interfaces these mechanisms offer.

Whereas said mechanisms are very generic and allow a variety of business models to be implemented, roaming cannot be described without making assumptions about deployment options that result from particular business models. It is not possible to capture all possible deployment options in this chapter, which is therefore informative in its entirety.

Two scenarios are described that show how roaming can be implemented for both interactive and broadcast mode of operation.

The placement of the Rights Issuer (RI) with respect to the Service Operation Centre (SOC) and the Customer Operation Centre (COC) is the main difference between the two described scenarios: in the first scenario, SOC combines Service Distribution Management (DIST Mgmt) and RI, in the second scenario, COC combines Service Subscription Management (SUB Mgmt) and RI.

In the description of the scenarios, the following important concepts need further explanation:

Locality	<p>A SOC and a COC can be considered local to each other, if</p> <ul style="list-style-type: none"> – the COC's complete pricing information is included in the SOC's service guide – the COC keeps its complete purchasing information current through "bulk downloads" – (if the COC includes an RI) the COC keeps the keys current through "bulk downloads" <p>Whereas Locality is not needed from a functionality point of view (a non-local COC can offer the same functionality as a local COC), it can be assumed that thanks to the better user experience and also less network traffic, Locality will be implemented at various degrees.</p>
Home COC	<p>The COC with which the owner of the device has a contractual relationship (aka. "Subscription"). This relationship allows that the owner is charged for the services consumed. In the case where the Home COC includes a Rights Issuer (RI), it can be assumed that the device is already registered and has a valid RI context.</p>
Local COC	<p>A COC that is local to a SOC.</p>
Home SOC	<p>The SOC which is local to the Home COC. In the case where the Home SOC includes an RI, it can be assumed that the device is already registered and has a valid RI context.</p>
Local SOC	<p>A SOC that is local to a COC.</p>
Visited SOC	<p>The SOC from which the device (at home or roaming) receives the Broadcast Services it wants to purchase.</p>

Note: The case where the visiting owner of a device enters a contractual relationship with a COC that is local to the Visited SOC is not considered roaming for the sake of this chapter (the case is identical to the local scenarios described in this chapter; "only" interoperability between devices and networks is needed to enable this functionality).

10.1 Deployment Option A (RI as part of SOC)

The first roaming scenario is based on the deployment option A, where the SOC combines DIST Mgmt and RI.

10.1.1 Local Scenario

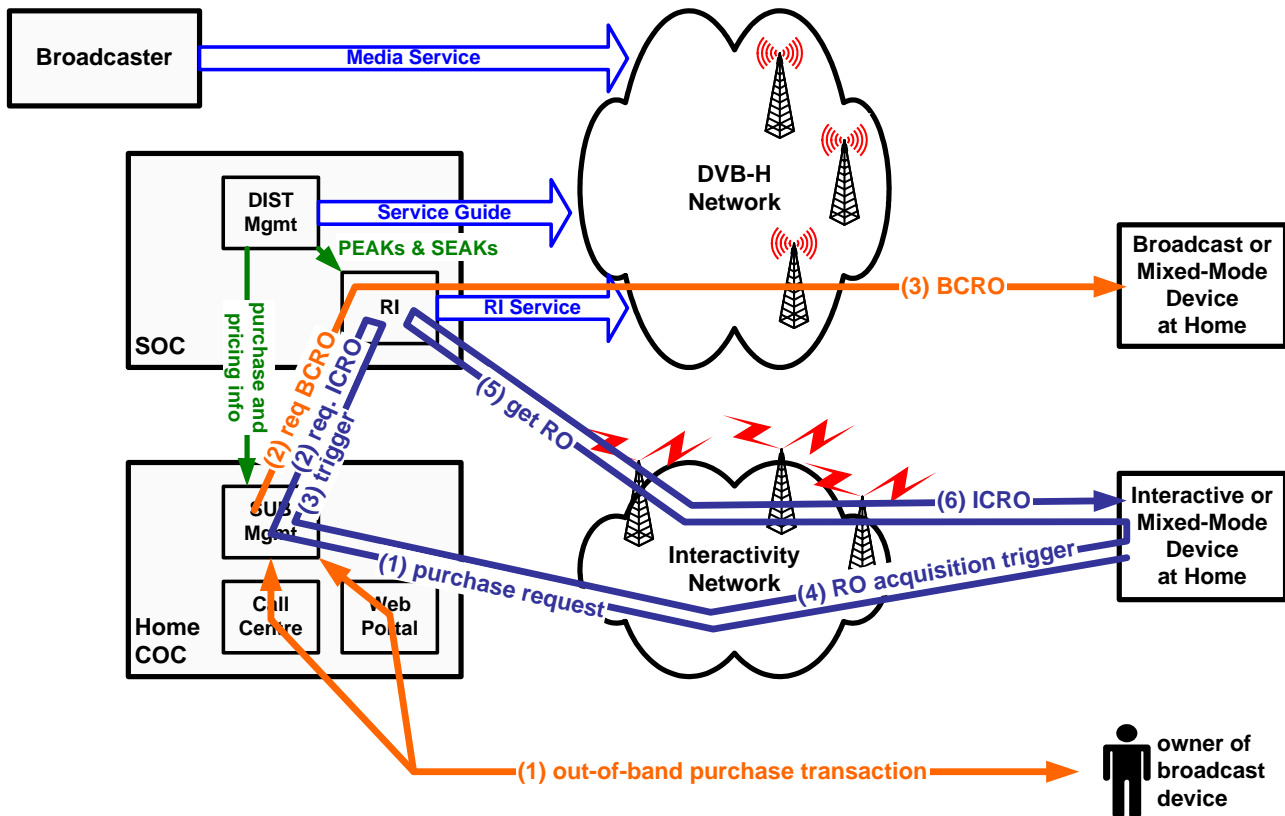


Figure 55: Deployment Option A (Combining DIST Mgmt and RI in SOC) – Local Scenario

In the **interactive mode of operation**, the device (1) issues a purchase request to the SUB Mgmt of the Home COC. If successful, the SUB Mgmt (2) requests the RI of the SOC to generate an ICRO related to the purchase, and (3) gets an RO acquisition trigger back, which it (4) forwards to the device. The device (5) uses the trigger to request the ICRO directly from the RI, which (6) sends the RO to the device.

In the **broadcast mode of operation**, the owner of the device (1) performs an out-of-band purchase transaction with the SUB Mgmt of the Home COC (e.g. through a call centre, or a web portal). If the transaction is successful, the SUB Mgmt (2) requests the RI of the SOC to generate and broadcast the BCRO related to the purchase. If this is successful, the RI (3) broadcasts the BCRO on the Ad-hoc RI Stream for a while, and then continues repeating it in the Scheduled RI Stream during the whole lifetime of the purchased item.

In the **mixed-mode of operation**, the device (1, in the interactive scenario) issues a purchase request to the SUB Mgmt of the Home COC. If successful, the flow can continue with the interactive mode of operation, or the broadcast mode of operation described above (2, in the respective scenario).

10.1.2 Roaming Scenario

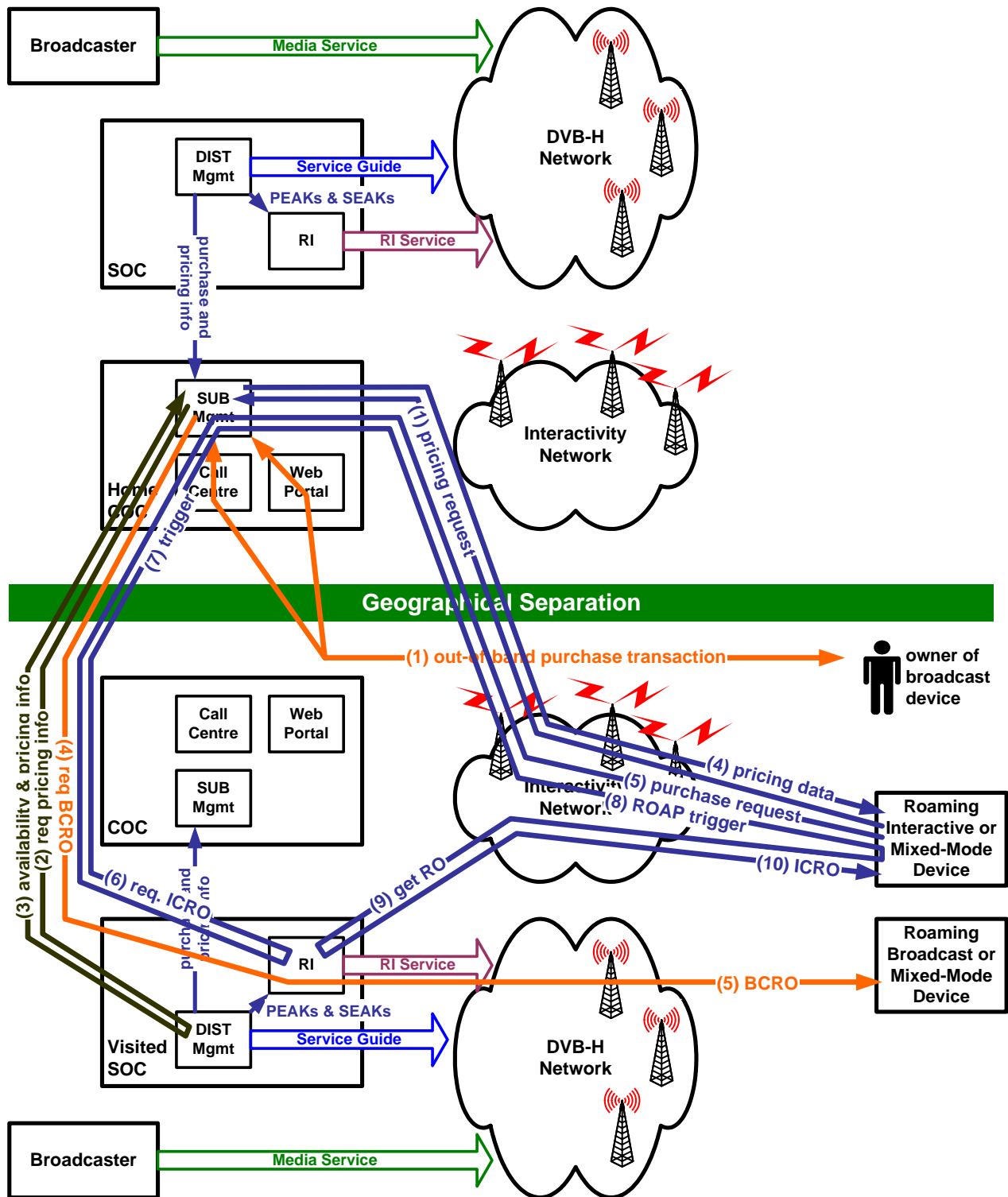


Figure 56: Deployment Option A (Combining DIST Mgmt and RI in SOC) – Roaming Scenario

In the **interactive mode of operation**, the device (1) first issues a pricing request to the SUB Mgmt of the Home COC, which is now remote to the Visited SOC. The SUB Mgmt of the Home COC (2) requests the availability and pricing information from the Visited SOC, which (3) returns it through the COC (4) to the device. Only now has the device all the information available that enables it (5) to make a purchase request to the SUB Mgmt of the Home COC. If successful, the SUB Mgmt (6) requests the RI of the Visited SOC to generate an ICRO related to the purchase, and (7)

gets an RO acquisition trigger back, which it (8) forwards to the device. The device (9) uses the trigger to request the ICRO directly from the RI of the Visited SOC, which (10) sends the RO to the device.

In the **broadcast mode of operation**, the owner of the device (1) performs an out-of-band purchase transaction with the SUB Mgmt of the Home COC (e.g. through a call centre, or a web portal). During the purchase transaction, the SUB Mgmt (2) requests the availability and pricing information from the Visited SOC, which (3) returns it for immediate use in the purchase transaction. If the transaction is successful, the SUB Mgmt of the Home COC (4) requests the RI of the Visited SOC to generate and broadcast the BCRO related to the purchase. If this is successful, the RI of the Visited SOC (5) broadcasts the BCRO on the Ad-hoc RI Stream for a while, and then continues repeating it in the Scheduled RI Stream during the whole lifetime of the purchased item.

In the **mixed-mode of operation**, the device (1) first issues a pricing request to the SUB Mgmt of the Home COC, which is now remote to the Visited SOC. The SUB Mgmt of the Home COC (2) requests the availability and pricing information from the Visited SOC, which (3) returns it through the COC (4) to the device. Only now has the device all the information available that enables it (5) to make a purchase request to the SUB Mgmt of the Home COC. If successful, the flow can continue with the interactive mode of operation (6), or the broadcast mode of operation (4) described above.

10.1.3 Conclusion to Deployment Option A

The advantage of this deployment option is that the programme encryption and authentication keys (PEAKs) and service encryption and authentication keys (SEAKs) never leave the Visited SOC.

The differences between the local and the roaming scenario are:

- the additional request for pricing information (the Home COC might cache this information for subsequent usage).

The needs for non-local communication between Visited SOC and Home COC in the roaming scenario are the following:

- retrieval of pricing information
- request for ICRO creation and retrieval of the corresponding acquisition trigger (interactive mode of operation)
- request for BCRO creation and broadcasting within an RI service (broadcast mode of operation)

10.2 Deployment Option B (RI as part of COC)

The second roaming scenario is based on the deployment option B, where the COC combines SUB Mgmt and RI.

10.2.1 Local Scenario

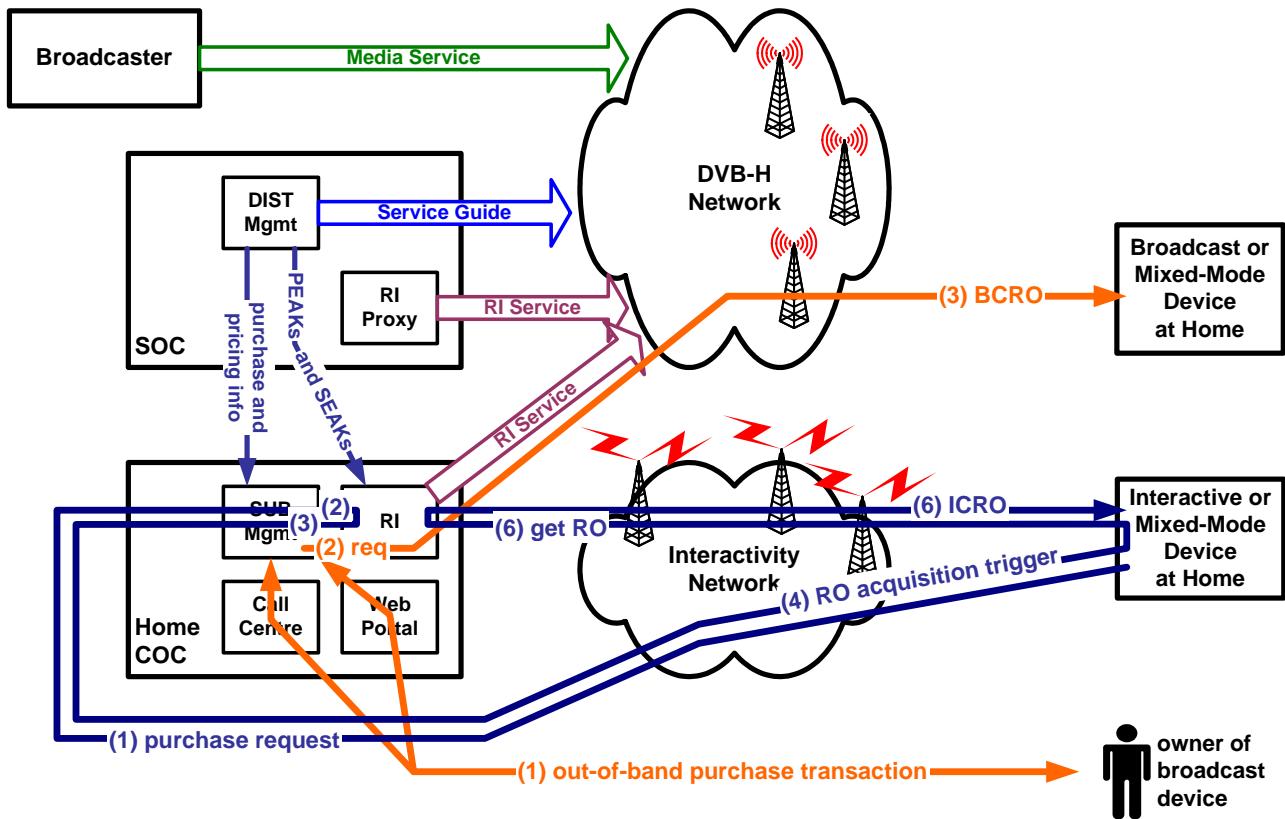


Figure 57: Deployment Option B (Combining SUB Mgmt and RI in COC) – Local Scenario

In the **interactive mode of operation**, the device (1) issues a purchase request to the SUB Mgmt of the Home COC. If successful, the SUB Mgmt (2) requests its own RI to generate an ICRO related to the purchase, and (3) gets an RO acquisition trigger back, which it (4) forwards to the device. The device (5) uses the trigger to request the ICRO directly from the RI, which (6) sends the RO to the device.

In the **broadcast mode of operation**, the owner of the device (1) performs an out-of-band purchase transaction with the SUB Mgmt of the Home COC (e.g. through a call centre, or a web portal). If the transaction is successful, the SUB Mgmt (2) requests its own RI to generate and broadcast the BCRO related to the purchase. If this is successful, the RI (3) broadcasts the BCRO on the Ad-hoc RI Stream for a while, and then continues repeating it in the Scheduled RI Stream during the whole lifetime of the purchased item.

In the **mixed-mode of operation**, the device (1, in the interactive scenario) issues a purchase request to the SUB Mgmt of the Home COC. If successful, the flow can continue with the interactive mode of operation, or the broadcast mode of operation described above (2, in the respective scenario).

10.2.2 Roaming Scenario

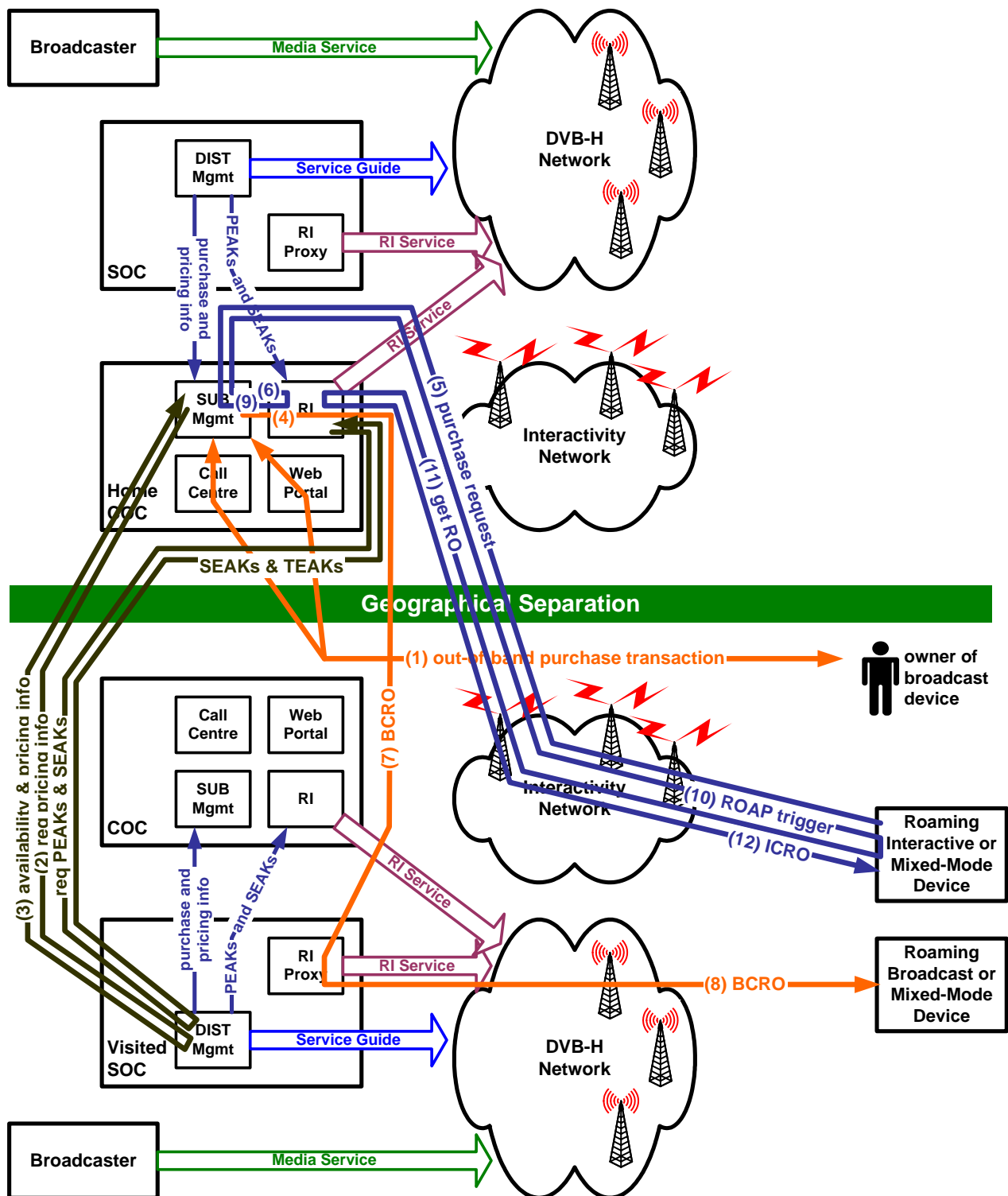


Figure 58: Deployment Option B (Combining SUB Mgmt and RI in COC) – Roaming Scenario

In the **interactive mode of operation**, the device (1, not shown in figure) first issues a pricing request to the SUB Mgmt of the Home COC, which is now remote to the SOC. The SUB Mgmt of the Home COC (2) requests the availability and pricing information from the Visited SOC, which (3) returns it through the Home COC (4, not shown in the figure) to the device. Only now has the device all the information available that enables it (5) to make a purchase request to the SUB Mgmt of the Home COC. If successful, the SUB Mgmt of the Home COC (6) requests its own RI to generate an ICRO related to the purchase. The RI of the Home COC (7, not shown in the figure) requests the

corresponding PEAKs and SEAKs from the DIST Mgmt of the Visited SOC, which (8, not shown in the figure) sends them back to the RI. The RI of the Home COC now generates the ICRO, and (9) returns an RO acquisition trigger back to the SUB Mgmt, which the SUB Mgmt (10) forwards to the device. The device (11) uses the trigger to request the ICRO directly from the RI of the Home COC, which (12) sends the RO to the device.

In the **broadcast mode of operation**, the owner of the device (1) performs an out-of-band purchase transaction with the SUB Mgmt of the Home COC (e.g. through a call centre, or a web portal). During the purchase transaction, the SUB Mgmt of the Home COC (2) requests the availability and pricing information from the Visited SOC, which (3) returns it, for immediate use in the purchase transaction. If the transaction is successful, the SUB Mgmt of the Home COC (4) requests the RI of the Visited SOC to generate and broadcast the BCRO related to the purchase. The RI of the Home COC (5, not shown in the figure) requests the corresponding PEAKs and SEAKs from the DIST Mgmt of the Visited SOC, which (6, not shown in the figure) sends them to the RI of the Home COC. The RI now generates the BCRO, and (7) requests the Visited SOC's RI Proxy to include the BCRO in its RI Service. The RI Proxy (8) broadcasts the BCRO on the Ad-hoc RI Stream for a while, and then continues repeating it in the Scheduled RI Stream during the whole lifetime of the purchased item.

In the **mixed-mode of operation**, the device (1, not shown in figure) first issues a pricing request to the SUB Mgmt of the Home COC, which is now remote to the SOC. The SUB Mgmt of the Home COC (2) requests the availability and pricing information from the Visited SOC, which (3) returns it through the Home COC (4, not shown in the figure) to the device. Only now has the device all the information available that enables it (5) to make a purchase request to the SUB Mgmt of the Home COC. If successful, the flow can continue with step (6) of the interactive mode of operation, or step (4) of the broadcast mode of operation described above.

10.2.3 Conclusion to Deployment Option B

The advantage of this deployment option is that the roaming device doesn't have to perform a RI registration when roaming, but that the ROs will be issued by the RI that is part of the Home COC.

This comes at the cost of:

- the Visited SOC having to make the programme encryption and authentication keys (PEAKs) and service encryption and authentication keys (SEAKs) available to the Home COC
- the Visited SOC having to operate an RI Proxy to run an RI service that will broadcast BCROs on behalf of the Home RI.

The differences between the local and the roaming scenario are:

- the additional request for pricing information that the Home COC makes from the Visited SOC (the Home COC might cache this information, for subsequent usage with other devices)
- in the local scenario, the Home COC has the choice to either use the SOC's RI Proxy, or to run its own SI Service (this is in principle also possible in the roaming scenario, but not desirable)

The reasons why there is a need for non-local communication between Visited SOC and Home COC in the roaming scenario are the following:

- retrieval of pricing information
- retrieval of PEAKs and SEAKs
- request for ICRO creation and retrieval of the corresponding acquisition trigger (interactive mode of operation)
- request for BCRO creation and broadcasting within an RI service (broadcast mode of operation)

10.3 Conclusion to Roaming Scenarios

The deployment and roaming scenarios presented in this chapter are made up from simple building blocks that can be flexibly combined. Whereas particular business models have been chosen to illustrate how "roaming" can be implemented, the building blocks are not limiting the deployment options.

Annex A (normative):

A.1 Security Considerations

A.1.1 Handling weak keys

(The responsible components in) the head-end architecture SHALL NOT use weak keys that will be used for messages in the IP Datacast network. At the time of this writing there are no specified weak keys for use in AES. This does not mean to imply that weak keys do not exist. If, at some point, a set of weak keys for AES are 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 certificate chain has been revoked or has expired, then the device is still allowed to use it for a short period of time (24 hours) during which the user has time to set the process in motion through which the device will receive a new certificate. This means that the user can enjoy the content he was entitled to consume straight away, at the expense of a slightly increased security risk of being able to use possibly compromised certificates for a somewhat longer time, i.e. 24 hours.

A device in broadcast mode SHALL implement the grace period mechanism.

- 1) The device checks periodically a particular or all RI context for expiration.
- 2) If a RI context is expired, the device displays a certificate expiry reminder for the associated RI context. The reminder notifies the user that the user needs to get a new certificate (of course in terms that a user can understand like “Call this number with this message please”)
- 3) Until this certificate expiry reminder is invoked the device will be rendered inoperable, but only in relation with the associated RI (context) as described below:
 - a. Accessing an ESG for purchase is still allowed.
 - b. 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.
- 4) Upon invoking the certificate expiry reminder the first time, the device starts the grace period via a secure timer, which will end 24 hours after it has been invoked. The device can use the invalid certificate until the secure timer expires.
- 5) If the secure timer (i.e. grace period) expires and the certificate chain has not been updated, the device will be rendered inoperable, but only in relation with the associated RI (context) as described below:
 - a. Accessing an ESG for purchase is still allowed.
 - b. 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 automatically schedule the certificate chain updates.

- 1) An update (powerup/powerdown) timeslot is programmed in which the RI will transmit the certificate chain. The timeslot may be obtained from the ESG. The device SHOULD parse the received ESG data to find a time at which it can receive a certificate chain update. Note that it may be the case that certificate chain updates are broadcast continuously. See section 7.8 for more details.

- 2) Upon power down before update the device may display a warning message that the device needs to update it's 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 section describes the status and error values for use 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 user¹⁴.

The service cannot continue due to an error.
 Please contact customer service at:
 XXXX-XXX-XXXXXXX

and notify the short UDN:
 XXXX XXXX
 with following errorcode
 XXX

An example dialogue showing an error

Figure 59: sample notification display

Note: The error codes should be displayed as a three digit decimal number. Refer to table 73 for an overview of possible error codes.

¹⁴ 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 specification. 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 (please note: the short UDN will only be displayed after the first registration, when that data is available for display).

Table 73: 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	0x11-0xFF	

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 offline 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 exclusively use it's interactive channel and not it's OOB channel.

ForceOobChannel indicates that the RI forces a mixed mode device to exclusively use it's OOB channel and not it's interactive channel.

A.3 Mapping of messages to DVB-H Time Sliced Bursts

This chapter specifies how the various key streams and other messages SHALL be mapped to DVB-H Time Slicing bursts. For definition of such a burst, see [EN 301 192].

A.3.1 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.

A.3.2 RI Service Channel

There are no requirements for the mapping of RI Services to DVB-H bursts.

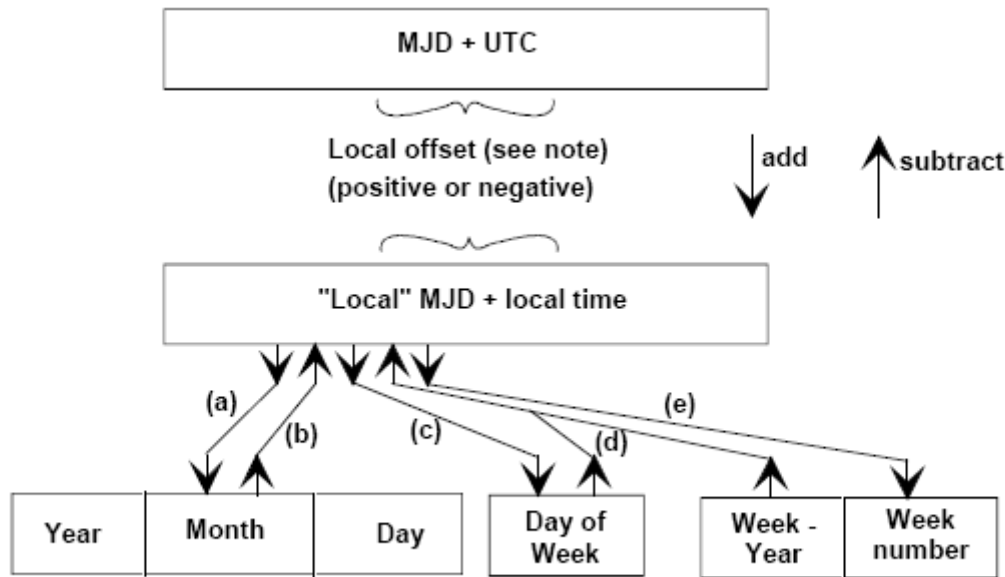
A.3.2.1 In-Band RI Stream

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.

A.4 Conversion between time and date conventions

(please note: this text has been copied from ETSI EN 300 468 V1.6.1)

The types of conversion which may be required are summarized in Figure 60.



NOTE: Offsets are positive for Longitudes East of Greenwich and negative for Longitudes West of Greenwich.

Figure 60: Conversion routes between Modified Julian Date (MJD) and Co-ordinated 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
MN	Week number according to ISO 2015 [21]
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
x	Multiplication
Y	Year from 1900 (e.g. for 2003, Y = 103)

a) To find Y, M, D from MJD

$$\begin{aligned}
 Y' &= \text{int} [(\text{MJD} - 15\,078,2) / 365,25] \\
 M' &= \text{int} \{ [\text{MJD} - 14\,956,1 - \text{int} (Y' \times 365,25)] / 30,6001 \} \\
 D &= \text{MJD} - 14\,956 - \text{int} (Y' \times 365,25) - \text{int} (M' \times 30,6001) \\
 \text{If } M' = 14 \text{ or } M' = 15, \text{ then } K &= 1; \text{ else } K = 0 \\
 Y &= Y' + K \\
 M &= M' - 1 - K \times 12
 \end{aligned}$$

b) To find MJD from Y, M, D

$$\begin{aligned}
 \text{If } M = 1 \text{ or } M = 2, \text{ then } L &= 1; \text{ else } L = 0 \\
 \text{MJD} &= 14\,956 + D + \text{int} [(Y - L) \times 365,25] + \text{int} [(M + 1 + L \times 12) \times 30,6001]
 \end{aligned}$$

c) To find WD from MJD

$$\text{WD} = [(\text{MJD} + 2) \bmod 7] + 1$$

d) To find MJD from WY, WN, WD

$$\text{MJD} = 15\,012 + \text{WD} + 7 \times \{ \text{WN} + \text{int} [(\text{WY} \times 1\,461 / 28) + 0,41] \}$$

e) To find WY, WN from MJD
 $W = \text{int} [(MJD / 7) - 2\,144,64]$
 $WY = \text{int} [(W \times 28 / 1\,461) - 0,0079]$
 $WN = W - \text{int} [(WY \times 1\,461 / 28) + 0,41]$
 EXAMPLE: MJD = 45 218 W = 4 315
 $Y = (19)82$ WY = (19)82
 M = 9 (September) N = 36
 D = 6 WD = 1 (Monday)

NOTE: These formulas are applicable between the inclusive dates 1900 March 1 to 2100 February 28.

A.4.1 local time offset

This 16-bit field contains the current offset time from UTC in the range between –12 hours and +13 hours at the area which 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. Please note that 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 follows:

Table 74: 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.5 Conversion of OMA DRM 2.0 content identifiers to binary content identifiers

[Note to the editor: This chapter sets out a number of requirements on the ESG and must be checked once the ESG specification is complete and available.]

The CID in OMA DRM 2.0 has the form of a URI. For a binary rights object as defined in section 6.3.4.2, 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 ESG	String
content_id	textual id as used in OMA DRM 2.0 and signalled in the ESG	String
content_type	“#P” for programmes and “#S” for services	String
cid-url	socID + content_type + content_id + “@”	String

The 64 bit binary content id BCI is then given by

$\text{base_BCI} = \text{HMAC-SHA1-64}(\text{cid-url})$

The extension_BCI is a 32 bit id (see program_CID_extension and service_CID_extension in the KSM).

The BCI is given by

$$BCI = (base_BCI \ll 32) | extension_BCI$$

A.6 Conversion of OMA X509PKIHash values to binary values

Values codes as roap:X509SPKIHash in OMA DRM 2.0 are SHA1-160 hashes. In OMA DRM 2.0 these values are presented as base64 encodings in a XML <hash> element as shown by example below:

```
<keyIdentifier xsi:type="roap:X509SPKIHash">
  <hash>aXENC+Um/9/NvmYKiHDLaErKofk=</hash>
</keyIdentifier>
```

If such a hash is carried in a binary object e.g. the BCRO or a KSM the 160-bit hash itself without the base64 encoding will be used.

A.7 Limits of the surplus_block()

Following examples show two possible cases: one keyset for standards addressing and one keyset with additional domain addressing.

A.7.1 Standard keyset

for standard addressing is keyset_block filled with:

- 1 UGK, 9 BGK, 1UDK, 1 UDF, 1 RIAK, 1 UDF.

Note: Max broadcast groupsize is 512, this is supported by 9 BGK.

Table 75: standard keyset with RSA block size 1024

value	variable	key size	key data	key size	key data	key size	key data
1024RSA size							
1UDF		40	40	40	40	40	40
1UGK		128	128	128	128	128	128
9BGK		128	1152	128	1152	128	1152
1UDK		128	128	128	128	128	128
1RIAK		128	128	128	128	128	128
0LDK		128	0	128	0	128	0
0SLDF		48	0	48	0	48	0
0LLDF		840	0	840	0	840	0
0TDK		128	0	128	0	128	0
13TLF overhead		7	181	7	181	7	181
keyset_block			1757		1757		1757
1SK		128	128	192	192	256	256
0PKCS overhead			0		0		0
sessionkey_block()		1024		1024		1024	
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 following results:

Table 76: standard keyset with other RSA block sizes

block size	SK size	keyset_block	room in sessionkey_block()	surplus block
2048	128	1757	1920	no
2048	192	1757	1856	no
2048	256	1757	1792	no
4096	128	1757	3968	no
4096	192	1757	3904	no
4096	256	1757	3840	no

A.7.2 Extended keyset

An extended keyset includes keys for standard addressing plus domain addressing. The keyset_block is filled with:

- 1 UGK, 9 BGK , 1UDK , 1 UDF , 1 RIAK, 1 UDF, 1 LDK, 1 SLDF, 1 LLDF (maximum size), 1 TDK.

Note: Max broadcast groupsize is 512, this is supported by 9 BGK.

Table 77: extended keyset with RSA block size 1024

value	variable	key size	key data	key size	key data	key size	key data
1024RSA size							
1UDF		40	40	40	40	40	40
1UGK		128	128	128	128	128	128
9BGK		128	1152	128	1152	128	1152
1UDK		128	128	128	128	128	128
1RIAK		128	128	128	128	128	128
1LDK		128	128	128	128	128	128
1SLDF		48	48	48	48	48	48
1LLDF		840	840	840	840	840	840
1TDK		128	128	128	128	128	128
17TLF overhead		7	219	7	219	7	219
keyset_block			2939		2939		2939
1SK		128	128	192	192	256	256
0PKCS overhead			0		0		0
sessionkey_block()		1024		1024		1024	
room in sessionkey_block() to use for keyset_block			896		832		768
(remainder of keyset_block in) surplus_block()			2043		2107		2171

Table 78: extended keyset with other RSA block sizes

block size	SK size	keyset_block	room in sessionkey_block()	surplus block
2048	128	1924	1920	1019
2048	192	1924	1856	1083
2048	256	1924	1792	1147
4096	128	1924	3968	no
4096	192	1924	3904	no
4096	256	1924	3840	no

Note: not yet included is the PKCS overhead in the sessionkey_block(), so surplus_block() will be a little larger.

A.8 Function F_{ZMB}

A.8.1 Definitions

The function for the ZMB system are 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 NK_i , the left child key is NK_{2i+1} and the right child key is NK_{2i+2} .

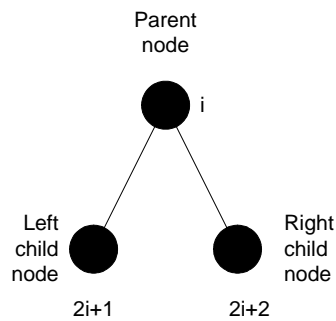


Figure 61: Node numbering

- b) **node / leaf key derivation function.** Given a parent key NK_i , the derived child keys NK_{2i+1} and NK_{2i+2} are calculated by encrypting a known constant using AES as shown in figure 62:

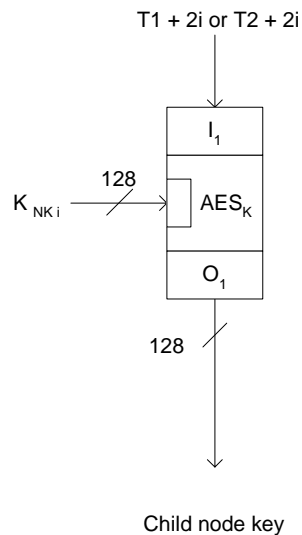


Figure 62: AES for key derivation

Explaining figure 62

step 1: Initialise the system with following values:

- Let $T1 = 0x01010101010101010101010101010101$
- Let $T2 = 0x02020202020202020202020202020202$

- Load K_{NK_i} (msblf), where $K_{NK_i} = BGK$ (in case of device) or $K_{NK_i} = \text{Root Key } NK_0$ (in case of RI).

step 2: Calculate the Left Child Node and Right Child Node from an Input Key BGK. Or, Given a parent key NK_i (i.e. Root Key or BGK), the derived child keys NK_{2i+1} (i.e. Left Child Node) and NK_{2i+2} (i.e. Right Child Node) are specified by following functions:

$$NK_{2i+1} = AES\{NK_i\}(2i+T1)$$

$$NK_{2i+2} = AES\{NK_i\}(2i+T2)$$

Note: AES compliant to [FIPS 197].

- 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 NK_0 that is known only to the rights issuer.
- leaf number calculation.** In a broadcast group for n devices, these n devices are associated with the leaf keys NK_{n-1} up to and including NK_{2n-2} . A device with position p (numbered from 0 to $n-1$) in the group is associated with key NK_{p+n-1} .
- determining the (BGK) 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
```

- 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 a DEK by concatenating the device revocation keys and using this concatenation as key for computing a MAC over the broadcast content identifier BCI as retrieved from the rights object:

$$DEK = HMAC - SHA1_{128}\{NK_{ex-1} \parallel NK_{ex-2} \parallel \dots \parallel NK_{ex-n}\}(BCI)$$

- DEK calculation.** The notation $HMAC_SHA1(k, s)$ is used to denote the computation of HMAC [RFC 2104] keyed by the key 'k' over the string 's' with SHA1 [FIPS 180.2] 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 that [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 frequently be the case in this algorithm. A device that has verified that its own key is not part of the exclusion key set, can calculate the DEK using the following algorithm:

```
function CalculateEncryptionKey
{
    Dek = 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
    Dek := HMAC_SHA1_128(Temp, Salt)
    return Dek
}

function CalculateNodeKey(i)
{
```

```

if NKi is in KeySet {
    return NKi from KeySet
}
else {
    parentkey = CalculateNodeKey( parent(i) )
    return AES{ parentkey } ( i )
}
}

```

A.8.2 Examples

A.8.2.1 Sample tree and keyset

Following picture shows a sample tree with keys for a device at position 7 (i.e. Device D0), as defined in subsection ‘e’ of A.8.1.

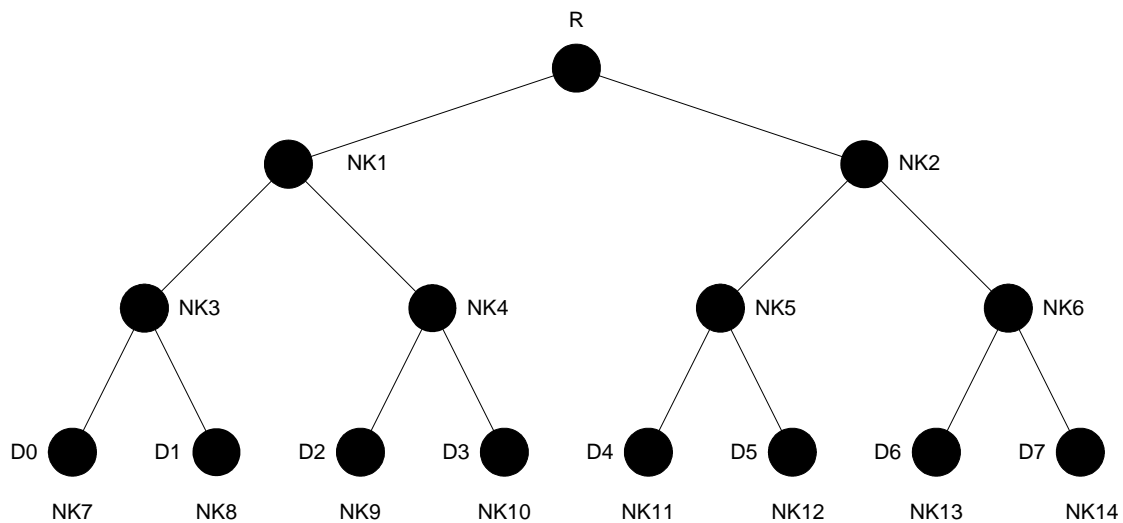


Figure 63: sample tree with correct node and device numbering.

E.g.1: Keyset for D0 = {NK2, NK4, NK8}

E.g.2: Keyset for D4 = {NK1, NK6, NK12}

E.g.3: To effectively disallow devices D1, D5 and D7 to access a certain asset, the rights issuer derives a DEK by concatenating the device revocation keys (NK_8 , NK_{12} and NK_{14}), and using this concatenation as key for computing a HMAC-SHA1-128 over the broadcast content identifier BCI as retrieved from the rights object:

$$DEK = \text{HMAC} - \text{SHA1}_{128}\{NK_8 \parallel NK_{12} \parallel NK_{14}\}(BCI)$$

A.9 Authentication

For a quick overview of the authentication “hierarchy” of the system refer to section 5.4.1.6.

A.9.1 Authentication for IPsec

IPsec can be used with authentication. In case of authentication with IPsec the authentication data will carry the TAS. The authentication mechanism will create the TAK from the TAS.

To obtain the encrypted traffic key material from the KSM the encrypted traffic key material is decrypted with the SEK or PEK:

$$TAS = D\{SEK\}(traffic_key_material)$$

or

$$TAS = D\{PEK\}(traffic_key_material)$$

The authentication key is generated from the authentication seed:

$$TAK = f_{auth}\{TAS\}(CONSTANT_KSM)$$

where:

$$CONSTANT_KSM = 0x04040404040404040404040404040404 \text{ (120 bit)}$$

Refer to A.9.4 for details on f-auth.

The TAK is used in the MAC generation / verification of the IPsec data. Refer to [RFC 2406] for details.

A.9.2 Authentication for KSMs

A key stream message can contain two MAC fields. The programme MAC field and the service MAC field. If only one MAC field would be 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) will 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 explained in subsequent sections). A service RO will contain service encryption and authentication keys (SEAK) and a programme RO will contain programme encryption and authentication keys (PEAK).

To obtain the PAS or TAS from the BCRO the SEAK/PEAK is decrypted with the DEK:

$$SAS = LSB_{128}(D\{DEK\}(SEAK))$$

$$PAS = LSB_{128}(D\{DEK\}(PEAK))$$

The authentication key is generated from the authentication seed:

$$SAK = f_{auth}\{SAS\}(CONSTANT_SAK)$$

$$PAK = f_{auth}\{PAS\}(CONSTANT_PAK)$$

where :

$$CONSTANT_SAK = 0x02020202020202020202020202020202 \text{ (120 bit)}$$

$$CONSTANT_PAK = 0x01010101010101010101010101010101 \text{ (120 bit)}$$

Refer to A.9.4 for details on f-auth.

The SAK or PAK is used in the MAC generation / verification of the KSM. The algorithm used to calculate the MAC field is HMAC-SHA1-96 according to [FIPS 198] and [RFC2104], using authentication keys of 160 bit in both cases.

A.9.2.1 Transport of SEAK and PEAK in OMA DRM 2.0 Rights Objects

The encryption keys and authentication keys (SEAK and PEAK) are transported in a ICRO by concatenating the encryption key and the authentication seed and then protecting the resulting field with AES_wrap [AES_WRAP].

encrypted_service_encryption_authentication_key =

$$E\{REK\}(SEAK) = AES_wrap\{REK\}(SEK << 128) \parallel SAS)$$

where SEK and SAS are both an AES key of 128 bits.

and encrypted_programme_encryption_authentication_key =

$$E\{REK\}(PEAK) = AES_wrap\{REK\}(PEK \ll 128) \parallel PAS)$$

where PEK and PAS are both an AES key of 128 bits.

A.9.2.2 Transport of SEAK and PEAK in BCROs

The encryption keys and authentication keys (SEAK and PEAK) are 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 =

$$E\{DEK\}(SEAK) = AES_CBC\{DEK\}(SEK \ll 128) \parallel SAS)$$

where SEK and SAS are both an AES key of 128 bits.

and encrypted_programme_encryption_authentication_key =

$$E\{DEK\}(PEAK) = AES_CBC\{DEK\}(PEK \ll 128) \parallel PAS)$$

where PEK and PAS are both an AES key of 128 bits.

A.9.3 Authentication of BCROs

BCROs contain one MAC field which is used to authenticate the message and to protect the integrity of the message.

The authentication key is generated from the RIAK:

$$BAK = f_{auth}\{RIAK\}(CONSTANT_BCRO)$$

where:

CONSTANT_BCRO = 0x03030303030303030303030303030303 (120 bit)

Note: To obtain the RIAK the device needs to have been equipped with a valid keyset. Refer to 6.4.3 for details.

Refer to A.9.4 for details on f-auth.

The BAK is used in the MAC generation / verification of the BCRO. The algorithm used to calculate the MAC field is HMAC-SHA1-96 according to [FIPS 198] and [RFC2104], using a authentication key of 160 bit.

A.9.4 General authentication mechanism

The function F-auth consists of several steps:

1. Denote by PRF{key}(text) as the AES-XCBC-MAC-PRF with output blocksize 128 bits as defined by IPsec WG in IETF. Please note:
 - Refer to [RFC 3566] for the AES-XCBC-MAC-PRF based key generation function.
 - Refer to [RFC 3664] for the requirement NOT to truncate the generated key material.
2. 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⁺ describes the function that outputs a pseudo-random stream of n blocks based on the inputs to a PRF as follows:

$$T1 = AES_XCBC_MAC_PRF\{AS\}(CONSTANT \parallel 0x01)$$

$$T2 = AES_XCBC_MAC_PRF\{AS\}(T1 \parallel CONSTANT \parallel 0x02)$$

....

$$Tn = AES_XCBC_MAC_PRF\{AS\}(T1 \parallel CONSTANT \parallel n)$$

where *AS* is the appropriate authentication seed (be it TAS, PAS, SAS or RIAK) and *CONSTANT* is the appropriate constant as described in preceding sections. 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 $PRF^*(\cdot)$ would be computed as:

$$T1 \parallel T2 = PRF^+\{K\}(S)$$

3. The 160 bit authentication key is taken from the generated key material as follows:

$$AK = MSB_{160}(T1 \parallel T2)$$

The generated authentication key is applied as described in preceding sections.

A.9.5 Authentication of token delivery response messages

Token delivery response messages contain one MAC field which is used to authenticate the message and to protect the integrity of the message.

The authentication key is generated from the RIAK:

$$TDRMAK = f_{auth}\{RIAK\}(CONSTANT_TDRM)$$

where:

CONSTANT_TDRM = 0x04040404040404040404040404040404 (120 bit)

Note: To obtain the RIAK the device needs have been equipped with a valid keyset. Refer to 6.4.3 for details.

Refer to A.9.4 for details on f_{auth} .

The TDRMAK is used in the MAC generation / verification of the token delivery response message. The algorithm used to calculate the MAC field is HMAC-SHA1-96 according to [FIPS 198] and [RFC2104], using a authentication key of 160 bit.

A.10 Checksum algorithms

According to empirical research by [VERHOEF_1969] the likelihood of errors is expressed as:

nr	error	representation	relative likelihood in %
1	single substitution	$a \Rightarrow b$	60 to 95
2	single adjacent transpositions	$ab \Rightarrow ba$	10 to 20
3	twin errors	$aa \Rightarrow bb$	0,5 to 1,5
4	jump transpositions (Longer jumps are even rarer)	$acb \Rightarrow bca$	0,5 to 1,5
5	phonetic errors (phonetic, because in some languages the two have similar pronunciation, e.g., thirty and thirteen)	$a0 \Rightarrow 1a$ where $a=\{2,...,9\}$	0,5 to 1,5
6	adding or omitting digits		10 to 20

Key:

$a < > b$, while *c* can be any decimal digit.

The most common errors are therefore errors 1, 2 and 6. Error 6 is easily detected. Following sections explain a method to detect other errors.

A.10.1 UDN checksum

Definition

The checksum on the UDN is calculated by $F\text{-UDN}$

We use codes over \mathbb{Z}_p , the integers modulo p , where $p=11$. That is to say, codewords are strings with entries from $\{0,1,\dots,p-1\}$. We consider codes of length n defined by r parity equations: a string (c_1, c_2, \dots, c_n) with elements from \mathbb{Z}_p is a codeword if and only if it satisfies the following equations:

$$\text{for } i = 1, 2, \dots, r, \sum_{j=1}^n a_j^{(i)} c_j \equiv 0 \pmod{p}$$

We now describe a $[20,17]$ code, that is defined over 20 symbols from \mathbb{Z}_{11} using the three following check equations as described in the matrix H_3 below:

Take $n=17$, $r=3$ and $p=11$. We consider the code defined by the $r=3$ following check equations:

$$\begin{aligned} 0*c_1 + 1*c_2 + 0*c_3 + \dots + 1*c_{18} &= 0 \pmod{11} \\ 1*c_1 + 0*c_2 + 1*c_3 + \dots + 1*c_{19} &= 0 \pmod{11} \\ 10*c_1 + 1*c_2 + 9*c_3 + \dots + 1*c_{20} &= 0 \pmod{11} \end{aligned}$$

In other words, a string $(c_1, c_2, \dots, c_{20})$ 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 H_3 :

	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15	n16	n17	n18	n19	n20
H3	1	0	1	0	1	0	1	0	1	0	1	2	3	4	5	7	8	1	0	0
	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 $r = (r_1, r_2, \dots, r_{20})$ satisfies the three parity check equations.

Encoding can for example be done as follows: Choose c_1, c_2, \dots, c_{17} in any way. If we define

$$\begin{aligned} c_{18} &= - (1*c_1 + 0*c_2 + 1*c_3 + \dots + 8*c_{17}) \pmod{11} \\ c_{19} &= - (0*c_1 + 1*c_2 + 0*c_3 + \dots + 7*c_{17}) \pmod{11} \\ c_{20} &= - (10*c_1 + 1*c_2 + 9*c_3 + \dots + 8*c_{17}) \pmod{11} \end{aligned}$$

then $(c_1, c_2, \dots, c_{20})$ is a codeword. We can view c_{18} , c_{19} and c_{20} as parity check digits. Note that we may restrict c_1, c_2, \dots, c_{17} to be any of the numbers $0, 1, 2, \dots, 9$. Any of the three parity check digits can be '10'. This '10' can be represented by an alphanumeric character different from $0, 1, \dots, 9$, for example X or Z.

Decoding is done by:

$$\begin{aligned} s_{18} &= (1*c_1 + 0*c_2 + 1*c_3 + \dots + 1*c_{18}) \pmod{11} \\ s_{19} &= (0*c_1 + 1*c_2 + 0*c_3 + \dots + 1*c_{19}) \pmod{11} \\ s_{20} &= (10*c_1 + 1*c_2 + 9*c_3 + \dots + 1*c_{20}) \pmod{11} \end{aligned}$$

Summarizing, the code defined with H_3 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 $ab \Rightarrow ba$ and a substitution is $a \Rightarrow b$.

Example:

Note: 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
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

 line for C18 & S18
 line for C19 & S19
 line for C20 & S20

coding

checkdigit = -sum(n1..n17) mod 11

C18

8	0	6	0	8	0	0	0	2	0	5	6	6	36	25	42	56
0	5	0	2	0	7	0	1	0	1	0	3	4	27	20	36	49
80	5	54	4	64	21	0	4	12	5	20	15	14	90	15	12	56

C19

C20

9

10

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(n1..n18 or n19 or n20) mod 11

S18

8	0	6	0	8	0	0	0	2	0	5	6	6	36	25	42	56	9	0	0
0	5	0	2	0	7	0	1	0	1	0	3	4	27	20	36	49	0	10	0
80	5	54	4	64	21	0	4	12	5	20	15	14	90	15	12	56	0	0	2

S19

S20

0

0

0

Please take note that the value '10' of checksum digit C19 can be represented by an alphanumeric character different from {0,1,...,9}, for example X or Z.

A.10.2 ARC checksum

Definition:

The checksum on the ARC is calculated by $F\text{-SDN}$

Take $n=12$, $r=2$ and $p=11$. We consider the code defined by the $r=2$ following check equations:

$$8*c_1 + 8*c_2 + 6*c_3 + \dots + 1*c_{11} = 0 \text{ (modulo 11)}$$

$$3*c_1 + 6*c_2 + 4*c_3 + \dots + 1*c_{12} = 0 \text{ (modulo 11)}$$

In other words, a string $(c_1, c_2, \dots, c_{12})$ 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 H^1 :

	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12
H1	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 $r = (r_1, r_2, \dots, r_{12})$ satisfies the two parity check equations.

Encoding can for example be done as follows: choose c_1, c_2, \dots, c_{10} in any way. If we define

$$c_{11} = - (8*c_1 + 8*c_2 + 6*c_3 + \dots + 4*c_{10}) \text{ modulo } 11$$

$$c_{12} = - (3*c_1 + 6*c_2 + 4*c_3 + \dots + 4*c_{10}) \text{ modulo } 11$$

then $(c_1, c_2, \dots, c_{12})$ is a codeword. We can view c_{11} and c_{12} as parity check digits. Note that we may restrict c_1, c_2, \dots, 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 alphanumeric character different from 0,1,...,9, for example X or Z.

Decoding is done by:

$$c_{11} = (8*c_1 + 8*c_2 + 6*c_3 + \dots + 1*c_{11}) \text{ modulo } 11$$

$$c_{12} = (3*c_1 + 6*c_2 + 4*c_3 + \dots + 1*c_{12}) \text{ modulo } 11$$

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 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 $ab \Rightarrow ba$ and a substitution is $a \Rightarrow b$.

Example:

Note: 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
	3	6	4	2	6	8	2	1	2	4	0	1
	line for C11 & S11											
	line for C12 & S12											
coding	checkdigit = $-\text{sum}(n1..n10) \bmod 11$											
C11	8	48	36	0	80	35	18	4	0	4		
C12	3	36	24	0	48	56	6	1	0	4		
codeword	1	6	6	0	8	7	3	1	0	1	9	9
decoding	checkdigit = $+\text{sum}(n1..n11 \text{ or } n12) \bmod 11$											
S11	8	48	36	0	80	35	18	4	0	4	9	0
S12	3	36	24	0	48	56	6	1	0	4	0	9

A.11 RSA signatures under PKCS#1

RSA signatures are made as described by the implementation guidelines of [PKCS #1] v2.1: RSA Cryptography Standard, *RSA Laboratories, June 14, 2002*.

The scheme is RSA + SHA1. There are two choices described in the [PKCS#1] 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, this specification will also use RSASSA-PSS to sign the binary messages for broadcast mode of operation.

A.12 C-style types

Following abbreviated types are used in the document:

type name	description	remark
bslbf	bit serial leftmost bit first	
mjdutc	modified julian date UTC	
uimsbf	unsigned integer most significant bit first	

All fields marked as reserved for future use SHALL be set to the value 0, when not used.

All fields marked as reserved SHALL be set to value 0, and never to any other value.

A.13 Tag Length Format for keyset_block

A.13.1 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>] <length> <keyset_item>

Following values are defined and SHALL be used:

tag values:

This is a 4 bit field (bslbf) indicating the tag that uniquely identifies the keyset item.

Table 79: defined tag values

Keyset_item	Tag (b)	remark
UGK	0000	
BGK	0001	
UDK	0010	
UDF	0011	
LDK	0100	
SLDF	0101	shortform_domain_id
LLDF	0110	
RIAK	0111	
TDK	1000	
reserved for future use	1001-1111	not used in this version of the spec

Note:

- The keyset items SHALL be included in the order of the table above.
- The keyset SHALL include only one instance of the following keys: UGK, UDK, UDF, RIAK and TDK.
- If included the BGKs (8 or 9) SHALL follow in fashion BGK1..n.
- The keyset MAY include zero or more domain sets (LDK, SLDF, LLDF). If included the SLDF SHALL follow the LDK it belongs to, followed by the optional LLDF that belongs to the aforementioned SLDF.

clarifier (optional):

This is a 10 bit field (bslbf) can be used to indicate the following possible values:

- in case the preceding <tag> value indicates a BGK, this field represents the position of a BGK in the Fiat Naor tree.
- in case the preceding <tag> value indicates a LLDF this field represents the length on the LLDF.

describing the use of the clarifier field for position of BGK:

If keyset_item == 001 (i.e. BGK) then the optional field “clarifier” SHALL indicate the position of the BGK as a node in the [FIAT NOAR] tree. When $m = \text{groupsize}$, then $n = 2 \log(m)$, where n is number of BGKs in tree. Possible positions for the BGKs in the tree are $2^{(n+1)} - 1$. Therefore parameter “position” is expressed with 10 bits to express 1023 nodes in a tree. First MSB left will be used as binary indicator to indicate if the BGK 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 and leafs SHALL be numbered according to following picture 64:

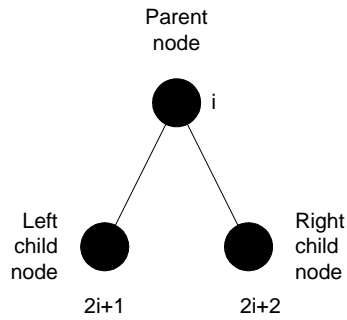


Figure 64: node numbering

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

describing the use of the clarifier for length of LLDF:

If LLDF is included the optional field “clarifier” describes the variable length of the LLDF in bits, as described in A.13.3.

length values:

This is a 3 bit field (bslbf) indicating the length of a keyset item.

Table 80: 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	SLDF
reserved for future use	101-111	not used in this version of the specification

Note: In case of the LLDF there is no extra length field, since the length value is indicated by the clarifier.

A.13.2 TLF examples

E.g.1: A 5 byte Eurocrypt address implementing the UDF will be coded like:

<0011> <011> <UDF>

E.g.2: A 48 bits SLDF address will be coded like:

<0101> <100> <SLDF>

E.g.3: A LLDF address of 105 bytes will be coded like:

<0110> <1101001000> <LLDF>

E.g.4: A 128 but AES key implementing the UGK will be coded like:

<0000> <000> <UGK>

E.g.5: A 128 bit AES key implementing the BGK on node position NK5 in figure 63 will be coded like:

```
<0001> <00000000101> <000> <BGK>
```

E.g.5: A 128 bit AES key implementing the BGK on node position NK7 (i.e. D0) in figure 63 will be coded like:

```
<0001> <10000000001> <000> <BGK>
```

A.13.3 LLDF 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. 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 are translated into bytes as follows:

Longest OMA DRM 2.0 domain identifier encoded as bytes is $6 \times 17 + 3$ bytes = 105 bytes.

Shortest domain identifier is 4 bytes.

A.14 Message_tag overview

The messages that are defined in this specification SHALL use following message_tag values:

Table 81: message_tag overview

message name	message_tag	described in section
BCRO()	0x20	6.3.4
device_registration_response()	0x01	6.4.3.7.2
re_register_msg()	0x11	6.4.3.7.3
update_ri_certificate_msg()	0x12	6.4.3.7.4
update_drmtime_msg()	0x13	6.4.3.7.5
update_contact_number_msg()	0x14	6.4.3.7.6
domain_registration_response()	0x02	6.4.4.4.1
domain_update_response()	0x03	6.4.4.4.2
join_domain_msg()	0x15	6.4.4.4.3
leave_domain_msg()	0x17	6.4.4.4.6
token_delivery_response()	0x30	6.4.5.4.1

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 section 6.4.3.3. in the way specified in this section. The hash function used here is one of the four secure hash functions from [SCHNEIER], 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 before mentioned restriction, is represented with a 14 bit uimbsf 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 uimbsf 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 section 6.4.5.4.1, 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 the next figure.

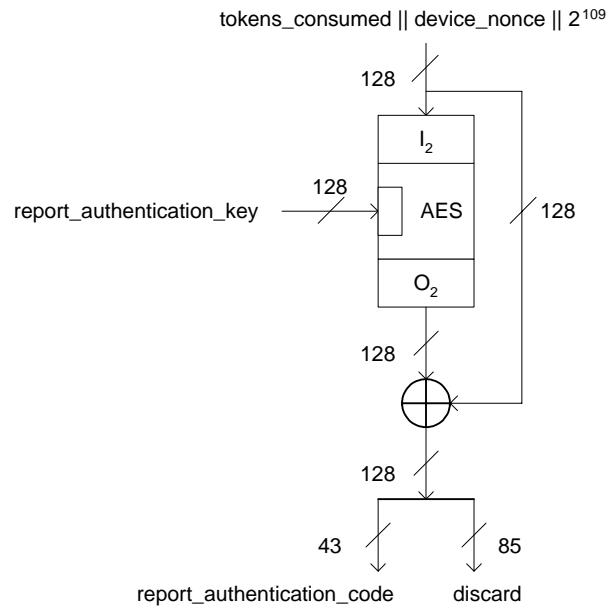


Figure 65: computation of the report_authentication_code

A.16 Management of tokens by RIs and devices

A.16.1 Token management by RIs

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 section 6.4.5.4.1 and the token reporting protocol, see section 6.4.5.3. The next sections describe how these tools can 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.16.1.1 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.16.1.2 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 tokens¹⁵.

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

1. 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.
2. The RI waits for the reception of a token consumption message.
3. 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 2, otherwise continue with step 4.
4. For reasons explained in section A.16.1.3, 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.
5. The RI bills the user for the amount of post-paid tokens consumed with a maximum equal to the credit-limit.
6. 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.
7. Go to step 2.

Note that 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 further that 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.16.1.3 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 1 in the previous section. The device will report the actual consumption of tokens that will be delivered to it in step 1 and in all steps 6. However, at the time of executing step 1, 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 section 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 `credit_limit` have been consumed by the device. Hence step 4 in section A.16.2.

A.16.1.4 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

¹⁵ Note that although a broadcast device can only display the token consumption message to the user and must rely on the user to report this message to the RI, the wording in this section is as if the device does the reporting.

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 e.g. when the previous option turns out to be expensive, is that the RI performs the actions described in section A.16.1.5 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.16.1.5 Stopping the post-paid token business model

When at a certain point of time, the user informs the RI that he does no longer wish 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 2 possible implementations of token consumption registration.

A device with a simple implementation of token consumption registration would do the following:

1. 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.
2. 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.

3. 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.
4. When a device creates a token consumption report, it uses a device_nonce which 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.
5. 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.
6. 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
7. The device SHALL stop with the execution of the above actions stop 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 Conformance Table

The follow table summarizes the mandatory and optional requirements for different types of devices, and for broadcasters and Rights Issuers.

Note that this table indicates whether support for each method is mandated. Please see the referenced chapter for details of whether the use of a method is mandated in specific circumstances, or whether alternatives are available.

	Reference	Device with Interactivity channel only	Device with Broadcast channel only	Mixed-mode device
Interactive mode registration	5.3.2	M	-	M
Broadcast mode registration	5.3.3	-	M	M
Mixed-mode registration	5.3.4	-	-	M
Addressing a complete group	5.6.2.1	-	M	M
Addressing a privileged subset	5.6.2.2 & 5.6.3	-	M	M
Addressing a unique device	5.6.2.3	-	M	M
Addressing a local domain	5.6.2.4	-	M	M
IPsec stream delivery	6.1.1	M	M	M
IPsec message authentication	6.1.1.4	M	M	M
SRTP stream delivery	6.1.2	M	M	M
SRTP message authentication	6.1.2.3	M	M	M
Key stream delivery	6.2	M	M	M
Timestamp field in KSM	6.2.1.3	M	M	M
Programme key layer in KSM	6.2.1.3	M	M	M
Service key layer in KSM	6.2.1.3	M	M	M
Access criteria in KSM	6.2.1.3	M	M	M
Parental rating in KSM	6.2.1.3	M	M	M
Permissions category in KSM	6.2.1.3	M	M	M
Access criteria descriptors in KSM	6.2.1.3	O	O	O
KSM authentication	6.2.1.3	M	M	M
Contents of service and programme RO	6.3.1, 6.3.2	M	-	M
Use of "datetime" constraint in service and programme ROs	6.3.1, 6.3.2	M	-	M
Scheduling of RO renewals according to datetime constraints	6.3.1	O	-	O
Common constraints for all assets in one service RO	6.3.1	-	-	-
Post-acquisition usage rules in service or programme RO according to [OMA-DRM-REL]	6.3.1, 6.3.2	M	-	M
Precedence of KSM permissions over RO permissions	6.3.1, 6.3.2	M	-	M
ICRO delivery with ROAP [OMA-DRM-DRM]	6.3.3	M	-	M
BCRO delivery	6.3.4	-	M	M
Parsing of BCRO	6.3.4.2	-	M	M
Support for asset object	6.3.4.2.1	-	M	M
Support for permission object	6.3.4.2.2	-	M	M
Support for action object	6.3.4.2.3	-	M	M
play action	6.3.4.2.3	-	M	M
display action	6.3.4.2.3	-	M	M
execute action	6.3.4.2.3	-	M	M
print action	6.3.4.2.3	-	M	M
export action	6.3.4.2.3	-	M	M
access action	6.3.4.2.3	-	M	M
Support for constraint objects	6.3.4.2.4	-	M	M
Support for count_constraint_descriptor	6.3.4.2.4.1	-	M	M
Support for timed_count_constraint_descriptor	6.3.4.2.4.2	-	O	O
Support for datetime_constraint_descriptor	6.3.4.2.4.3	-	O	O
Support for interval_constraint_descriptor	6.3.4.2.4.4	-	O	O
Support for accumulated_constraint_descriptor	6.3.4.2.4.5	-	O	O
Support for individual_constraint_descriptor	6.3.4.2.4.6	-	O	O
Support for system_constraint_descriptor	6.3.4.2.4.7	-	M	M
Support for metering_constraint_descriptor	6.3.4.2.4.8	-	O	O
Notify offline requests with detection of vocal errors	6.4.3.3	-	M	M
Inform device on forced actions	6.4.3.5 & 6.4.3.7.3 & 6.4.4.4.3 & 6.4.4.4.5	-	M	M

	Reference	Device with Interactivity channel only	Device with Broadcast channel only	Mixed-mode device
Device identification per UDN	6.4.3.6	-	M	M
Broadcast device registration data	6.4.3.7.2	-	M	M
Broadcast RI certificate update	6.4.3.7.4	-	M	M
Broadcast DRM time update	6.4.3.7.5	-	M	M
Broadcast contact number update	6.4.3.7.6	-	M	M
Joining a local domain	6.4.4	-	M	M
Leaving a local domain	6.4.4	-	M	M
Local domain registration data	6.4.4.4.1	-	M	M
Token handling	6.4.5	O	O	O
Reception of Rights Issuer Services	7	-	M	M
Broadcast of Rights Issuer Services	7	-	-	-
Provision of RI Service Schedule	7.2	-	-	-
Scheduling of RI Service reception	7.2	-	O	O
Purchase via interactivity channel	8.1	O	-	O
Support for all defined transactions if purchase via interactivity channel is supported	8.1	M	-	M
Support for join-domain trigger in ROAP registration response	8.1.1.5	M	-	M
Support for additional join-domain trigger in purchase response	8.1.1.6	M	-	M
Random distribution of renewal requests over time	8.1.1.7	M	-	M
Support for HTTP over the interactivity channel	8.1.2	M	-	M
Support for HTTPS over the interactivity channel	8.1.2	M	-	M
Signing of XML messages using RSASSA-PSS [PKCS#1]	8.1.3	M	-	M
Canonicalization of XML messages according to [XML_XCAN]	8.1.3	M	-	M
Announcement of device capabilities in XML requests	8.1.3.1.2	M	-	M
Usage of previously announced device capabilities if they are not included in a request	8.1.3.1.2	O	-	O
Usage of DVB platform ID as socID	8.3.1	-	-	-
Support for all purchase transactions if Purchase URL is announced in service guide	8.3.2	-	-	-
Support for out-of-band transactions if Portal URL is announced in service guide	8.3.2.	-	-	-
Inclusion of COC data in service guide for each COC with whom the SOC has an agreement	8.3.2	-	-	-
Inclusion of purchase availability information in service guide for local COCs	8.3.2, 8.3.7	-	-	-
Inclusion of price information in service guide for local COCs	8.3.2, 8.3.7	-	-	-
Handling weak keys	A.1.2	-	-	-
Handling OCSP grace period	A.1.3	-	M	M

Key:

M: Support is mandatory

O: Support is optional

-: Not applicable

Annex B (informative):

B.1 Service Purchase Scenarios

Section B.1.1 describes several service purchase scenarios.

B.1.1 Free-to-Air (unscrambled)

An unscrambled Free-to Air service does not use service protection as defined in this specification. If there is no Key Stream Message signalled for a service, it means that the service is an unscrambled Free-to-Air service.

B.1.2 Free-to-View (scrambled)

A Free-to-View service uses service protection as defined in this specification. The ROs associated with a Free-to-View service are sent to each registered device free of charge. The RI should schedule the sending of ROs associated with a Free-to-View service such that registered devices can always access Free-to-View service, except perhaps for a short time after device switch-on or device registration.

B.1.3 Subscription-based services

When a user wants to subscribe to a service, he performs a subscription transaction. The way a subscription transaction and subsequent actions may be carried out depends on which channel is used.

B.1.3.1 Using the Interactivity Channel

When an Interactivity Channel is available, the subscription transaction may be carried out over the Interactivity Channel, e.g. by filling in forms on a Web page. In this subscription transaction, the user somehow indicates this service and his device to the RI and possibly a payment method. After successful completion of this subscription transaction, the RI sends to the device of the user the ROs that govern the access to the service that the user subscribed to.

The delivery of ROs over the Interactivity Channel is specified in section 6.3.3.

B.1.3.2 Using the Broadcast Channel

When there is no Interactivity Channel available, the user will have to perform the subscription transaction using a different communication channel, e.g. the telephone. During the subscription transaction, the user communicates the following to the Customer Operation Centre of the Service Provider:

- The service(s) the user wants to subscribe to, possibly in the form of a Human Friendly Purchase ID. A Human Friendly Purchase ID is a Service ID suitable for humans to communicate and it is unique at least within the domain of a particular Service Provider. The Service Provider has to map Human Friendly Service IDs to Service IDs.
- The Human Friendly Device ID, either in the form of the shortform_udn or longform_udn(), as specified in section 6.4.3.7.2, or a Domain ID, as specified in section 6.3.4.2 and 6.4.4.
- A payment method.

After successful completion of this subscription transaction, the RI of the Service Provider sends to the device of the user the ROs that govern the access to the service(s) that the user subscribed to.

The delivery of ROs over the Broadcast Channel is specified in section 6.3.4.

B.1.4 One-Off Purchases

In a One-Off purchase, one typically purchases the right to view a certain service not for the full duration of that service, but only for a limited amount of time, e.g. the time that a particular movie is being broadcast.

In this specification, it is an option to divide a service into several programmes. If a service is divided into several programmes, the access control can be on a programme by programme basis, see e.g. section 5.4.1.2.

There are 2 variations for a One-Off purchase:

- PPV: the user pays for a complete programme.
- IPPV: the user pays for part of a programme, because he decided to do the One-Off purchase when the programme already had started.

The difference in between the above two can be in the pricing.

The process of a One-Off purchase is equal to the process of a subscription to a service as defined in section B.1.3, except that the ROs sent to the device of the user contain PEAKs instead of SEAKs (see also section 5.4.1.2).

B.1.5 Bundling with a mobile subscription

The protection technology defined by this specification does not offer any specific features for bundling mobile and IPDC subscriptions, nor does it put any requirements on whether a subscription is bundled or not with a mobile subscription.

B.1.6 Online and Offline control of content

This specification offers a transparent channel for protected content to be downloaded to the device. The possibilities are indicated in section 4..

B.1.7 Billing and Charging types

B.1.7.1 On line

'On line' indicates that the requested rights to consume a service are purchased with a direct connection between the authorisation server and the device (e.g. through interactivity channel).

An example of how an RO could be purchased on line is shown in section B.1.3.1. Independent of the way an RO is purchased, section B.1.3 and section B.1.4 show ways how the purchased ROs for a service (B.1.3) or part thereof in the form of a programme, see section B.1.4, can be transmitted to the device.

The consumption of content can also be paid for using tokens stored in the device, see section B.1.7.5. This section also shows how tokens can be purchased on line.

B.1.7.2 Off line

'Off line' indicates that the requested rights to consume a service are purchased without a direct and connected link between the authorisation server and the device.

An example of how an RO could be purchased off line is shown in section B.1.3.2. Independent of the way an RO is purchased, section B.1.3 and section B.1.4 show ways how the purchased ROs for a service (B.1.3) or part thereof (B.1.4) can be transmitted to the device.

The consumption of content can also be paid for using tokens stored in the device, see section B.1.7.5. This section also shows how tokens can be purchased off line.

B.1.7.3 Pre-Paid

'Pre-paid' indicates that the requested rights to consume a service or programme are charged before this service or programme is delivered and is available for consumption. The examples of consumption purchases in section B.1.3 and section B.1.4 are examples of pre-paid consumption if the sending of ROs commences after the user has been charged for these ROs.

Pre-paid can also mean that the content is purchased without any ongoing subscription with the RI. The example of the consumption of content using tokens in section B.1.7.5 is an example of this type of pre-paid purchasing model if the tokens are delivered to the device after the user has been charged for these tokens.

Note that the device always has to be registered with the RI of the Customer Operation Centre as specified in section 5.5, otherwise no ROs of any kind can be sent to the device. However, whether the Customer Operation Centre also wishes to obtain personal information of the user is outside the scope of this specification.

B.1.7.4 Post-Paid

'Post-paid' indicates that the requested rights to consume a service or programme are charged after this service or programme is delivered and is available for consumption. The examples of consumption purchases in sections B.1.3 and section B.1.4 are examples of post-paid consumption if the sending of ROs commences already before the user has been charged for these ROs.

Post-paid can also mean that the content is purchased without any ongoing subscription with the RI. The example of the consumption of content using tokens in section B.1.7.5 is an example of this type of post-paid purchasing model if the tokens are delivered to the device before the user has been charged for these tokens.

Yet another example of post-paid consumption in section B.1.7.5 is that a device consumes contents using tokens up to a certain limit for the maximum number of tokens and up to a certain date/time limit and that the device has to report the actual consumption in tokens before this date/time limit.

Note that the device always has to be registered with the RI of the Customer Operation Centre as specified in section 5.3, otherwise no ROs of any kind can be sent to the device. However, whether the Customer Operation Centre also wishes to obtain personal information of the user is outside the scope of this specification.

B.1.7.5 Tokens

'Token': A consumption unit used for purchasing services. To avoid on-line purchase of service, a token may e.g. be purchased or granted in advance, in which case it would typically be stored in the terminal.

An RI can send tokens to a device as specified in section 6.4.5.

The RI can send tokens to a device:

- Free of charge in order e.g. to pre-load a device with tokens in a post-paid purchasing model.
- After a user has ordered tokens using an Interactivity Channel. Billing of the tokens may be before or after delivery of the tokens.
- After a user has ordered tokens using another channel (e.g. by a phone call). Billing of the tokens may be before or after delivery of the tokens.

When sending tokens to a device free of charge, as indicated above, the RI may indicate that the device has to report the actual consumption of tokens to the RI before a certain date/time, see section 6.4.5.3. The device can only continue the consumption of tokens after the set date/time limit when it has received a new date/time limit from the RI. Note that RIs may wish not to support this type of token consumption for devices which do not have an Interactivity Channel, because the user cannot be forced to report the actual use of the last period.

Whether or not a device needs to use tokens for the consumption of content and if so how many, is indicated using the metered constraint, as specified in section 6.3.4.2.4.8. The metered constraint can be present in an RO (see section 6.3.4) as well as in the KSM (see section 6.2.1).

B.1.8 Bundles of Services

Bundles of services can be handled using the tools in this specification in e.g. the following ways:

- When a user subscribes to a bundle of services, he will be sent one RO per service with the SEK and SES of that service.
- When a user subscribes to a bundle of services, he will be sent one RO with the SEKs and SES-es of all services from the bundle.

- Any combination of the above.

B.1.9 Free Preview

Services may offer a “free preview” to users. Various scenarios for the offering of free previews and possible implementations are informatively described below. However, free previews may be implemented in any way that is compliant with this specification.

B.1.9.1 Free-to-Air Preview

The simplest form of free preview is to allow any user to watch a particular service without restriction for some period of time.

This can be implemented by making a part of the service “free-to-air” and broadcasting it unencrypted, as described in chapter 5.1.2.1.

B.1.9.2 Free Preview with Rights Objects

Free preview could be implemented in the same way as free-to-view service (see chapter B.1.1), where the service encrypted and a key stream is also broadcast. A Rights Object is delivered enabling users to receive the preview service, or perhaps all preview services of a particular Rights Issuer. The key stream and the Rights Object will carry a content ID which allows them to be matched.

B.1.9.3 Free Preview with Access Criteria

The above scheme could be used to restrict access to the preview service according to access criteria in the key stream, as described in chapter 6.2.1. For example, this could restrict access to registered users above a certain age.

B.1.9.4 Timed Preview

Users can be given the opportunity to receive a service for an arbitrary interval, duration, accumulated duration or with any other restriction that can be described using the OMA DRM 2.0 Rights Expression Language.

This can be implemented by delivering Rights Objects to all the users who are to be offered the preview. These Rights Objects would contain the necessary SEAKs or PEAKs required to receive the channel, along with the relevant restrictions. Note that the choice of supplying a SEAK or a PEAK and the relevant service or programme CID in the Rights Object will be influenced by how long the preview opportunity will last.

When a device receives a Rights Object that allows “timed preview”, it should indicate to the user that the preview is available.

These Rights Objects may be delivered to any or all registered users to fulfil any requirements a Rights Issuer may have.

B.1.9.5 Preview with Cryptographically-Enforced Period

Rights Issuers could divide a programme up into a series of sub-programmes, each with its own PEAK and CID. Rights Objects for the preview are made available to users containing only the PEAK and CID for the relevant sub-programme, so that the preview will last for the sub-programme length. The ESG will indicate a number of CIDs for the entire programme.

Alternatively, it may be the case that the preview lasts for exactly one programme, or more than one programme, in which case, the mapping of PEAKs and CIDs to programmes can be adjusted accordingly.

B.1.9.6 Advertising Previews via the ESG

In order that users can easily identify that a service offers a preview of some sort, it should be indicated in the ESG.

B.2 Operative Scenarios

B.2.1 Revocation

In certain cases, a need to revoke Device Private Keys may occur. Revoking a Device Private Key means that the certificate certifying the accompanying Device Public Key is included in the Certificate Revocation List (CRL).

When there is an Interactivity Channel, the device will obtain ROs using the ROAP protocol. This ensures that the RI will have to check the CRL any time it sends an RO. For a service requiring a monthly update of the RO, the RI will check the CRL monthly.

When there is no Interactivity Channel with a device, the RI will only use the Device Public Key during registration or re-registration.

The RI should check for the certificates of all devices that are registered on a regular basis whether or not they are on the CRL. If a certificate is on the CRL, the RI

- stops using all secrets (SEAKs, PEAKs, Broadcast Group Keys, etc.) that have been sent to the device to which that certificate belongs, and
- stops sending new ROs to the device to which that certificate belongs, and
- provides all other non-revoked devices that used these secrets with new UGKs, BGKs, ROs, using newly generated secrets.

B.2.2 Expiration

This specification defines two ways of operation, the interactive mode of operation and the broadcast mode of operation, see sections 5.5 and 5.6.

This specification makes use of functionality defined in OMA DRM 2.0 such as e.g. the ROAP protocol and ROs in the interactive mode of operation. In the interactive mode of operation, expiration is defined by OMA DRM 2.0, see [OMA_DRM_DRM].

In the broadcast mode of operation, this specification defines several objects that control expiration. This section highlights some of these.

B.2.2.1 Expiration of RI Context

During registration, a device receives the RI Context in a `device_registration_response()` message, see section 6.4.3.7.2. This message contains two objects which control expiration of the RI Context.

The first object is the certificate chain of the RI, as indicated in the `certificate_chain` field of the `device_registration_response()` message. Certificates in this certificate chain may contain expiration dates.

The second object is a time stamp, as indicated in the `time_stamp` field of the `device_registration_response()` message. The `time_stamp` field may be present in the `device_registration_response()` message.

Whenever any certificate in the certificate chain expires or when the `device_registration_response()` message expires based on the field `time_stamp`, the device stops using the access rights from all ROs that the device received and was able to access using the data (unique group key, broadcast group keys or unique device key) from this `device_registration_response()` message for making content received after the expiry time available for any purpose.

B.2.2.2 Expiration of a BCRO

The BCRO contains a field called `refresh_time`, see section 6.3.4.2. The date/time in this field indicates to the device that it is advised to try and acquire a newer version of the BCRO, because e.g. one or more end times of constraints in the BCRO are approaching. The `refresh_time` therefore does not indicate expiration of the BCRO.

B.2.3 Moving from one group to another

In order to save bandwidth when sending BCROs to devices, devices may be combined into groups. If several devices in one group require the same BCRO, then when using the group addressing feature of this specification, this BCRO only needs to be sent once and will be received by all addressed devices in the group, see section 5.6.2.

In order to optimise the bandwidth for sending BCROs to devices, it may be advantageous to move a device from one group to another.

Moving from one group to another can be done by sending a re-registration trigger to the device, see section 6.4.3.5. As part of the re-registration process, the RI provides the device with the credentials of the new group, using the `device_registration_response()` message, see section 6.4.3.7.2.

B.3 Scalability, Bandwidth Considerations

Sending out RO over the broadcast channel has a couple of implications. When an interactive device requires a new RO this RO can be delivered via the interactivity channel just in time and the RI can be sure that the RO has been delivered to the terminal. A broadcast device can't request a new RO when it needs one and the RI does not know whether the terminal has received an RO sent out via the broadcast channel. The terminal might have been switched off or the reception of the RO might have had errors. For these reasons the RO has to be transmitted repeatedly. This however requires a lot of bandwidth if every device is addressed directly as shown in figure 66 for a population of 500.000 devices. Even with a repetition rate of 60 minutes the data rate required for unique device addressing is more than 100 kBit/sec. Optimisations such as only sending RO in a certain broadcast cell for devices which are actually in that cell are not possible as the RI does not have that information. While this might be possible with interactive devices based on the mobile phone cell they are logged into this is not possible with broadcast devices as they don't register themselves with a broadcast/mobile phone cell. It is also not possible in Single-Frequency Networks. The figure 66 also shows how bandwidth can be saved by using group based addressing. In this case the same RO is send to a group of devices.

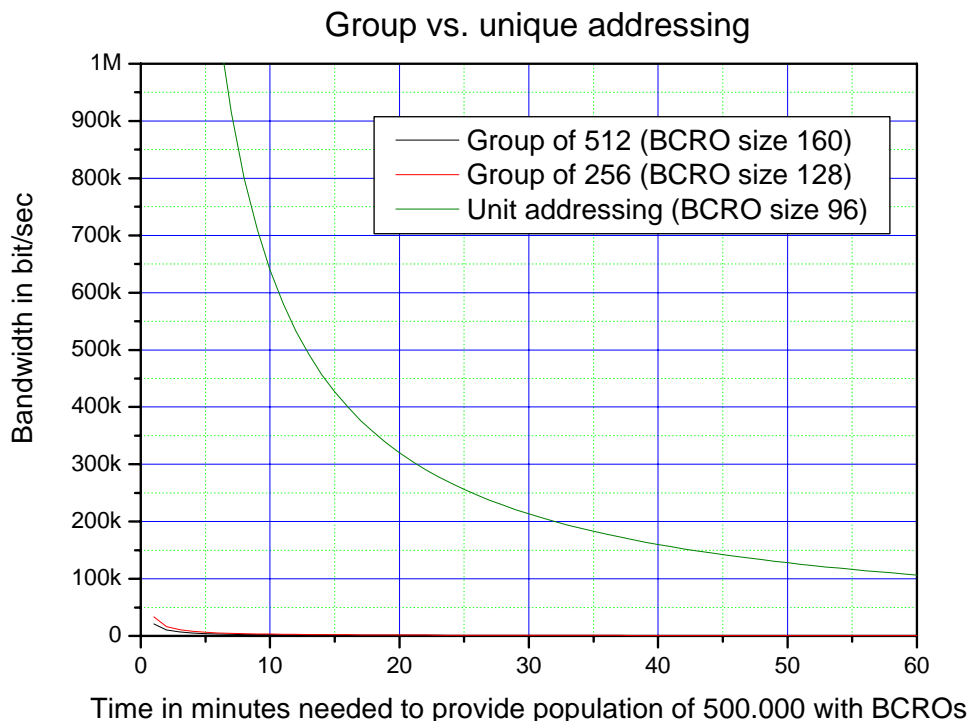


Figure 66: Bandwidth for RI service – unique addressing vs. group addressing

Figure 67 shows how a group size of 256 scales over different population sizes. The repetition rate (or time needed to send out all ROs) of the BCROs can be chosen by the broadcaster and is a trade-off between data rate required and the delay in terminals receiving the required ROs. In order to optimise the broadcast of ROs scheduling of ROs as defined

in section 7. can be used. Terminals can tune in, to receive the scheduled blocks of BCROs and hence don't have to rely on the repetitive broadcast of BCROs anymore. It is expected that blocks of BCROs will be schedule at night when free bandwidth is available. The repetition rate of BCROs in the RI service can therefore be reduced to save bandwidth.

Scheduling can also help to reduce power consumption as the terminal does not have to listen to the RI service constantly. It is likely that the scheduled RI Service will repeat BCROs periodically so that devices have several chances to receive them.

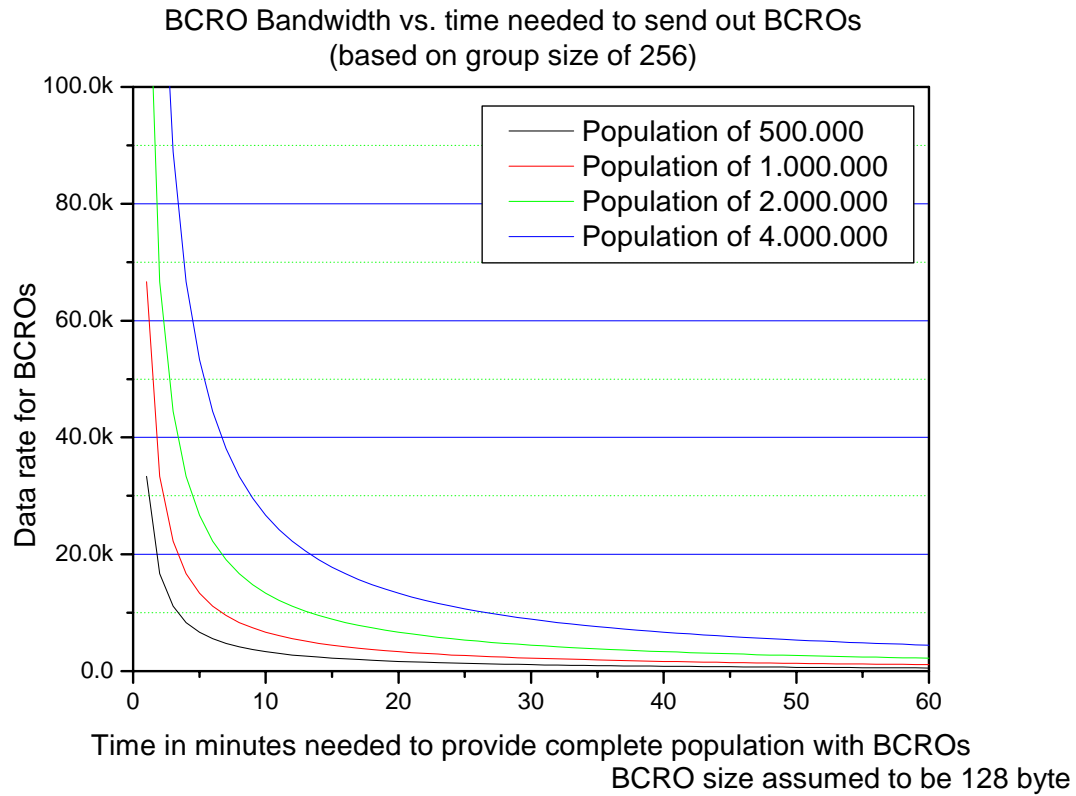


Figure 67: Data rate for RI service carrying BCROs

Figure 68 shows the influence smaller broadcast groups have on the data rate required for the BCROs.

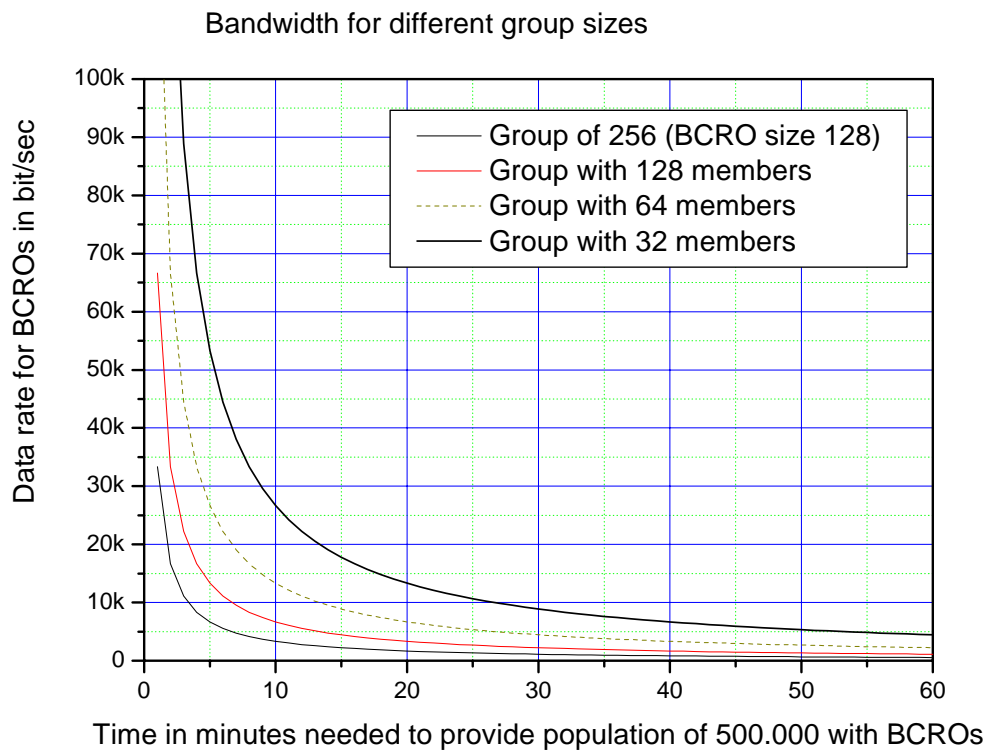


Figure 68: Data rate for RI Service carrying BCRO for different group sizes

If an RI has to deliver Rights Objects to both interactive and broadcast devices the RI may choose to deliver ROs to interactive devices over the interactivity channel or the broadcast channel, depending on which channel is more (cost) efficient or in order to reduce the load on a certain channel (broadcast or interactivity).

B.4 Use of the Interoperability Point

The interoperability point, or short i-point, allows multiple DRM implementations to be used simultaneously in an IPDC infrastructure. Rights to one and the same service can be sold via multiple Rights Issuers using multiple rights management systems and multiple DRM implementations. A Rights Issuer in this context does not imply an OMA DRM 2.0 Rights Issuer, it merely indicates an entity issuing Rights Objects independent of the DRM system used.

Given a service *A* from service provider *SPI*. *SPI* will encrypt the content and broadcast it together with a key stream. The base content ID of the service and the socID is signalled as part of the ESG and the service extension ID is signalled as part of the KSM.

The ESG will contain purchase information for each RI selling rights to a particular service or programme as shown in figure 69.

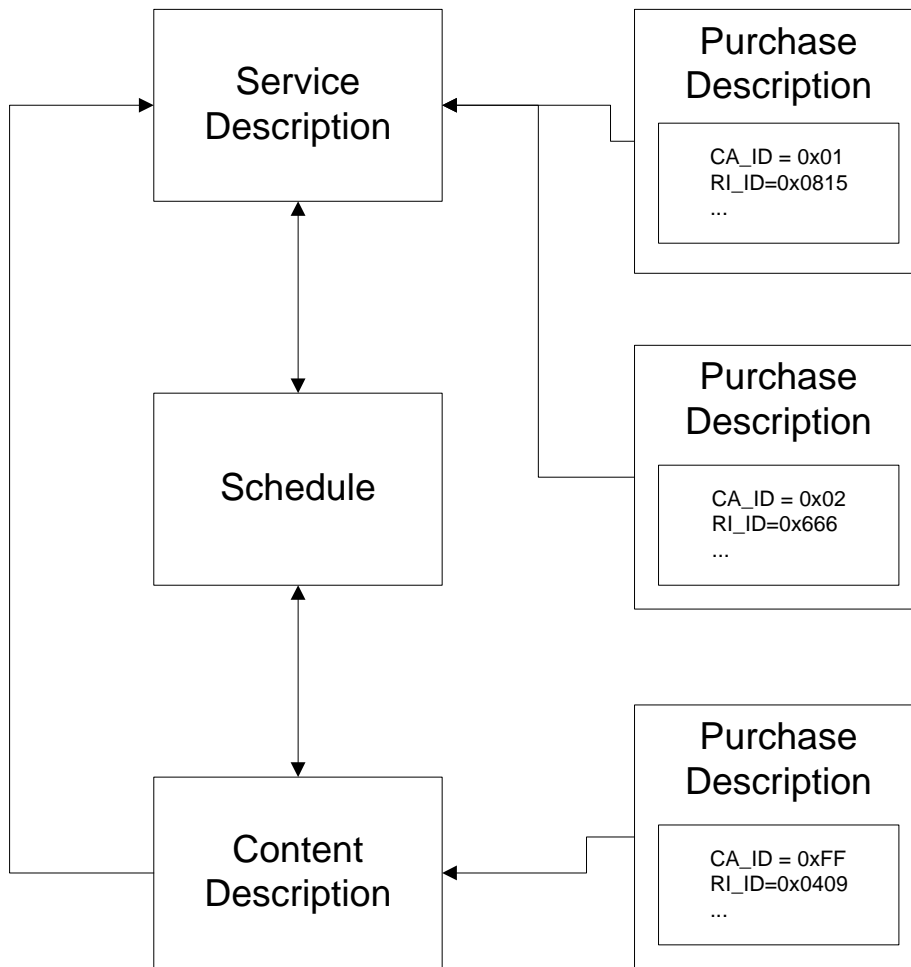


Figure 69: Use of multiple purchase descriptions in the ESG

The CA_ID is a unique ID identifying the DRM system used above the i-point. The values 0x00 and 0x01 are reserved. The value 0x01 indicates that the DRM system used above the i-point is the DRM system specified by this specification. The assignment of values to specific DRM system is outside of the scope of this specification.

The combination of the socID, content_id, content_type and the service extension ID0 uniquely identifies (see Annex A.5) the service and the keys used to encrypt the TEK. *SP/* provides details about the service (socID, content_id, content_type) together with key information (SEK/SAK or PEK/PAK, service_extension_ID) to all interested RIs. The RIs create either compliant ROs as defined in this specification or proprietary 'ROs' intended for other DRM implementations. These ROs get send to interested devices, either by means specified in this specification or by any other means in case of proprietary rights management systems. This specification does not specify proprietary ROs but in order to use the interoperability point, it needs to be possible to associate proprietary ROs to the unique service_CID and they have to carry the key material (SEK and SAK or PEK and PAK).

A device, which supports multiple DRM implementations is assumed to check with each implementation whether it has any ROs for a specific service_CID. A possible algorithm for checking multiple DRM implementations in a device is shown in figure 70.

When a user selects a service or programme the device acquires the socID, content_type (programme or service) and the service_base_ID from the ESG, together with the information on how to receive the KSM. Once the device has received the first KSM it can create the CID/BCI. Given the CID/BCI the device can query all DRM systems available in that device whether any of the DRM systems has a RO for that particular CID/BCI. The default DRM system to check should be the system specified in this specification. As an optimisation the device might only check with DRM systems listed in the purchase information of the ESG.

If none of the DRM systems in that device has a RO for that service/programme the access to that service/programme fails and the user could be notified that there are no entitlements to view that service/programme.

If at least one DRM system has a RO for that particular service/programme the DRM system is instructed to decrypt the KSM, check for its integrity and deliver the resulting TEK to the SRTP/IPsec implementation. Access permissions listed in the KSM for that particular RI_ID SHALL be adhered to by the DRM system.

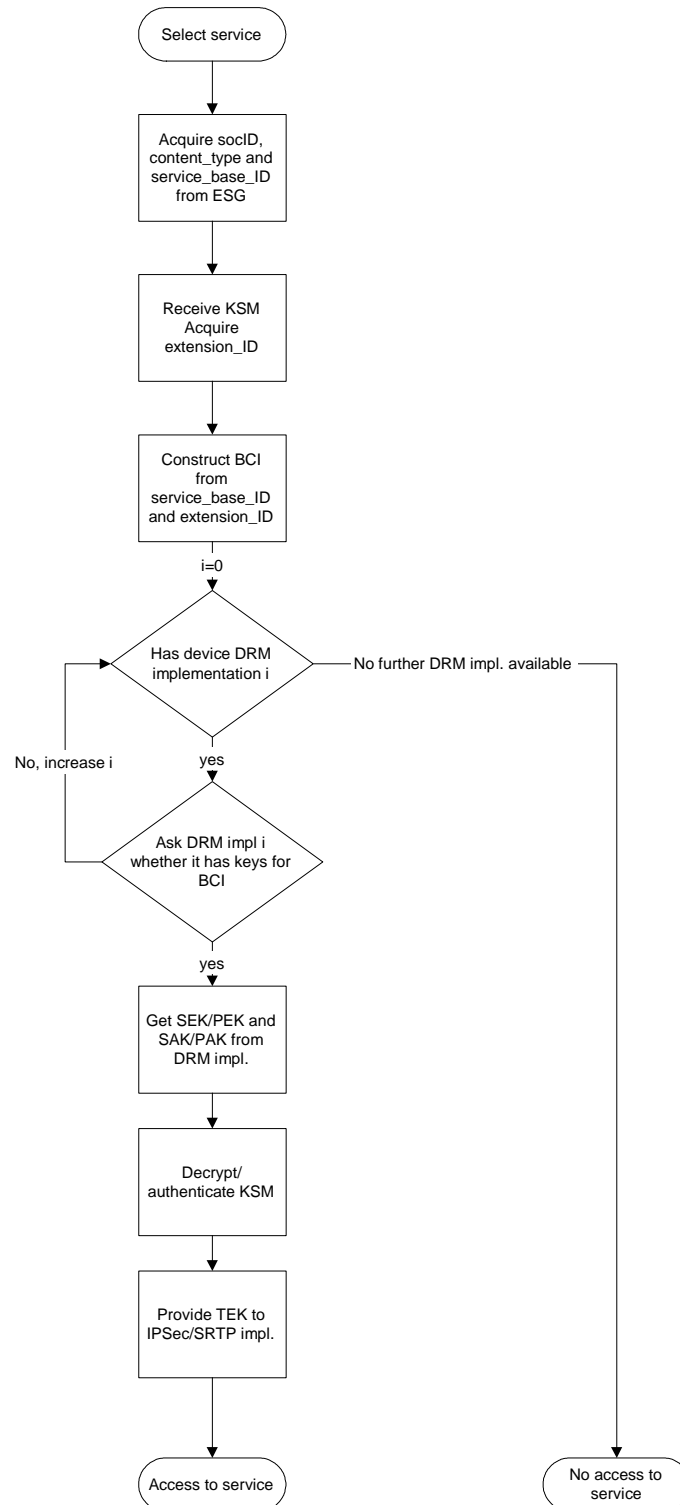


Figure 70: Process for service decryption when multiple DRM implementations are available in device

B.5 Security Considerations

This section explains the assumptions behind the security of the system. The system can be deployed either in interactive mode or broadcast mode, and can be used as the basis for service or content protection.

The host receiving the key stream messages, BCROs and IPsec or SRTP traffic must take care to protect itself from maliciously constructed key stream messages and rights objects. This is protection that must be afforded to the host and the user of the host. To protect the host the following conditions shall have to be true:

- Key stream messages shall not instantiate outbound IPsec security associations.
- An IPsec security association or SRTP cryptographic context instantiated based on a key stream message shall not interfere with any negotiated or user-configured IPsec security association.
- The amount of resources devoted to IPDC IPsec security associations and SRTP cryptographic contexts shall be limited so that a denial-of-service attack is not possible.
- The amount of key stream messages under processing shall be limited so that a denial-of-service attack is not possible against the host. This is essentially critical as the key stream message authentication key is stored in multiple systems and they may not be trustworthy.
- The amount of Rights Objects under processing shall be limited so that a denial-of-service attack is not possible against the host. This is especially critical as the authentication of Rights Objects may not be reliable (a symmetric key is used in the case of BCROs) and-or they may not be protected.
- For each IPsec security association and SRTP cryptographic context there must be one key stream that is authorized to modify the security association or cryptographic context. Modification attempts by unauthorized key streams shall be silently ignored.

To assure a level of protection for hosts against other malicious hosts in the system is based on the security of the authentication keys. This security is ultimately rooted in the tamper-resistance of the hosts holding the authentication keys. The following conditions shall have to be true:

- The BCRO authentication keys shall not be readily available in a receiving host.
- The key stream message authentication keys shall not be readily available in a receiving host.
- The IPsec security association authentication keys shall not be readily available in a receiving host.
- The SRTP cryptographic context authentication keys shall not be readily available in a receiving host.

Note that the authentication of key stream messages and BCRO objects is not reliable as the system does not guarantee status information about key validity, especially in broadcast mode. The message authentication codes of key stream messages and BCROs provide security in direct proportion to the weakest tamper-resistant hardware that has access to the used message authentication keys.

The protection that must be afforded for the decryption keys and the broadcast content is required for the security of the service and content protection. This protection will ultimately be rooted in tamper-resistance of the receiving hosts. If the system is deployed for service protection then the assets that must be protected from unauthorized access are the long-term secret keys on the device and short-term decryption keys for the broadcast data.

The keys are divided into two domains. Long-term and short-term. Long-term secrets are essentially secrets whose lifetime spans that of several rights objects. Short-term secrets are secrets that are changed frequently over time, at least changing per rights object, if necessary.

The long-term domain of keys is:

- BGKs, UDK and UGK.
- Intermediate keys derived when deriving a DEK from a set of BGKs.
- OMA DRM 2.0 decryption and signing keys.
- OMA DRM 2.0 domain keys.
- RIAK

The short-term domain contains all the other keys in the system.

The trustworthiness of a receiving host in terms of service protection can be indicated as a tuple of the following parameters:

1. Level of protection afforded long-term decryption secrets and asymmetric keys.
2. Level of protection afforded short-term secrets.
3. Level of protection afforded long-term symmetric authentication keys.

Parameter 1 defines the capability of a host to compromise content and decryption not intended for it. Parameter 2 defines the capability of a host to compromise content intended for it. Parameter 3 defines the capability of a host to pose as a legitimate rights issuer or broadcaster to other hosts in the system.

This specification does not as such define robustness requirements for implementations, but the recommendation is that long-term secrets should be afforded hardware-based protection, such that their storage and handling is performed using tamper-resistant hardware. It is probably a reasonable to rely on software protection (such as OS security) for short-term secrets, especially if the integrity of the handling software is assured.

If content protection is desired then the system must be hardened further. In this only authorized applications that heed post-acquisition and usage rules have access to decrypted keys and decrypted content. Note that SRTP does NOT authenticate any enveloping headers such as UDP ports. It is recommended that authentication be used in cases where content protection is used to prevent a malicious party from modifying the cipher text. This could result in redirecting decrypted traffic to another application. It is trivial to redirect traffic to another application if SRTP is used as SRTP does not authenticate the enveloping header. In this case it must be ensured that these unauthorized applications do not have access to the decryption keys. If IPsec is used, then it must be ensured that an unauthorized application does not receive the decrypted traffic.

If content protection is required then also the authentication keys must be secured at least as strongly as any decryption keys as they are used to prove authenticity and integrity of usage rules.

The trustworthiness of a receiving host for content protection can be measured in terms of the service protection trustworthiness parameters and the ability to limit decrypted content and keys to applications that heed the usage rules and security offered stored content.

The content if cached/stored on the local device may be stored in a multitude of ways, and it is beyond the scope of this specification to define how that is to be done. It is recommended that if content protection is desired the content be stored in such a manner that unauthorized applications cannot decrypt stored protected content or change the usage rules pertaining to stored protected content.

B.5.1 Threat Mapping

This section contains a simplified threat mapping of the system to highlight why the different cryptographic assets must be protected. This mapping is given in terms of assets, their function, the consequence of unauthorized use and possible mitigation techniques. The full transitive chain of consequences of the compromise of one asset leading to the compromise of another asset is not explicitly given here, but the cause-consequence relationships should be clear enough that the reader can work these out. This section does not consider how any decrypted content must be handled within the system, as that is a function of the robustness of the system, and outside the scope of this specification.

This section intends to provide input to the design of implementation, especially how to structure the manipulation of different secrets inside a possible device. This section does not aim to be a complete, normative or definitive treatment.

Asset	Device Private Key
Function	The device private key is used for registration and decryption of the BGKs, UGK and UDK addressed to this device. The key is part of a key pair, and the private key is used for decryption, with the device public key being used for encryption.
Consequence of unauthorized use	All keyset blocks (BGKs, UGK, RIAK and UDK) addressed to the device owning the key can be decrypted. These can be used to decrypt the fields in all BCROs addressed to the device encrypted using keys in these keyset blocks.
Mitigation	The device private key must be stored in a tamper-resistant hardware module. Only a

robust implementation of the non-interactive mode registration layer must be allowed use of it.

In theory, if there is a mechanism for detecting compromised device private keys, the corresponding public keys could be revoked using conventional techniques such as revocation lists.

Asset	OMA DRM 2.0 Device Private Keys
Function	These keys are used to authenticate OMA DRM 2.0 client-side messages and decrypt rights objects that are addressed to this device.
Consequence of unauthorized use	The OMA DRM 2.0 Rights Objects addressed to this device can be decrypted. A malicious party could create unauthorized signatures and pretend to the identity of the OMA DRM 2.0 implementation in the device that it could use these keys.
Mitigation	The private keys must be stored in a tamper-resistant fashion and the keys must only be accessible to a robust OMA DRM 2.0 implementation.
Asset	Unique Group Key
Function	This is a common symmetric key for all members in a group of devices in non-interactive mode. The group and the key are determined at the registration layer. This key is used to derive the DEK for when a BCRO is addressed to all the members in a non-interactive mode broadcast group.
Consequence of unauthorized use.	The DEK for all BCROs addressed to the corresponding group could be computed.
Mitigation	The unique group key should be stored in tamper resistant hardware and only be allowed to be used by a robust implementation of the non-interactive mode rights layer. Further, the HMAC_SHA1() hash function and the SALT input from the BCRO means that the DEK derived from the UGK does not reveal information about the UGK. The rights layer should be allowed access only to DEKs.
Asset	Broadcast Group Keys
Function	These keys are used to derive the DEKs for the broadcast groups in non-interactive mode. They are managed by the registration layer.
Consequence of unauthorized use	The DEKs for all the BCROs addressed to broadcast subgroups the recipient is a member of can be computed. If the BGKs of two different devices in the same group can be obtained, then all BCROs addressed to all subgroups in this group can be decrypted.
Mitigation	The broadcast group keys (BGKs) must be stored in tamper-resistant hardware and only be allowed to be used by a robust implementation of the non-interactive mode registration layer. The rights layer should be allowed access only to DEKs.
Asset	Unique Device Key
Function	The unique device key is used to form the DEK used to decrypt BCROs addressed to a single device in non-interactive mode.
Consequence of unauthorized use	BCROs addressed to this device would be decrypted.
Mitigation	The unique device keys (UDKs) must be stored in tamper-resistant hardware and only be allowed to be used by a robust implementation of the non-interactive mode registration

layer. The rights layer should be allowed access only to DEKs.

Asset OMA DRM 2.0 Domain Keys

Function These keys are used to decrypt rights objects that are addressed to members of an OMA DRM 2.0 domain.

Consequence of unauthorized use The OMA DRM 2.0 Rights Objects addressed to a domain can be decrypted.

Mitigation The OMA DRM 2.0 domain keys should only be accessible and usable by robust implementations of OMA DRM 2.0.

Asset RIAK

Function The Rights Issuer Authentication Key is used in non-interactive mode to derive the BA, which is used to authenticate BCRO objects. This is a symmetric key. This key is provided as part of the registration data.

Consequence of unauthorized use Unauthorized use of the RIAK allows a member of a broadcast group to pose as a rights issuer to other members of the same group. This could be used to create denial-of-service attacks or to allow an attacker to gain access to attack vectors that would otherwise have been protected.

Mitigation The RIAK should be stored in tamper-resistant hardware and it should only be usable and accessible by a robust implementation of the registration layer. Preferably the registration layer would only indicate whether a BCRO is authentic or not to the rights layer. If that is at all possible, then the registration layer should provide only a BAK to the rights layer.

Because the RIAK is a symmetric key, the rights layer of a device should not be able to compute MACs, only verify them.

Asset SEK

Function The service encryption key is the key provisioned in rights objects to devices. This key is used to encrypt TEKs and in-case a pay-per-view model is followed, PEKs.

Consequence of unauthorized use Unauthorized use of the SEK allows one to decrypt TEKs or PEKs one is not entitled to.

Mitigation The SEKs should not be revealed by the rights layer. Only decrypted TEKs should be revealed by the rights layer. The rights layer implementation should be robust.

Asset PEK

Function The program encryption key is the key provisioned in rights objects to devices. This key is used to encrypt TEKs in-case pay-per-view model is followed. This key may be also provisioned in a Key Stream Message, in which it would be encrypted using a SEK.

Consequence of unauthorized use Unauthorized use of the PEK allows one to decrypt TEKs one is not entitled to.

Mitigation The PEKs should not be revealed by the rights layer. Only decrypted TEKs should be revealed by the rights layer. The rights layer implementation should be robust.

Asset TEK

Function	The traffic encryption keys are used as the master key for the traffic layer (IPsec or SRTP). They are provisioned in Key Stream Messages and are encrypted using a SEK or a PEK.
Consequence of unauthorized use	The IPsec or SRTP traffic can be decrypted by an authorized party.
Mitigation	The TEK should be changed relatively frequently, on the order of once per a couple of seconds, if possible. The implementation handling the TEKs (the traffic layer) and the key stream layer, should be robust and not reveal the TEK to unauthorized parties nor allow its use for anything than incoming IPsec traffic. Backup/restore or import/export of TEKs by the traffic layer should not be allowed.
Asset	BAK
Function	The BAK is a key used to authenticate BCROs. It is derived from the RIAK.
Consequence of unauthorized use	Unauthorized use of the RIAK allows a member of a broadcast group to pose as a rights issuer to other members of the same group. This could be used to create denial-of-service attacks or to allow access to attack vectors that would otherwise have been protected.
Mitigation	<p>The BAK should not be stored, but should be derived as needed by the registration layer. Preferably the registration layer would only indicate whether a BCRO is authentic or not to the rights layer. If that is at all possible, then the registration layer should provide only a BAK to the rights layer. The RIAK used to create the BAK must never be revealed.</p> <p>Because the RIAK is a symmetric key, the rights layer of a device should not be able to compute MACs, only verify them.</p>
Asset	SAK
Function	The Service Authentication Key is derived from the encrypted service authentication seed inside a BCRO. This key is used to authenticate key stream messages.
Consequence of unauthorized use	Unauthorized use of the SAK allows a malicious party to create forged key stream messages. Possibly causing denial of service attacks, or allowing access to attack vectors that would not have been available otherwise.
Mitigation	The rights layer implementation should be robust, and it should not reveal the SAK to the key stream layer (or any other party). It should only provide a “authentic / not authentic” response for key stream messages.
Asset	PAK
Function	The Program Authentication Key is derived from the encrypted program authentication seed inside a BCRO. This key is used to authenticate key stream messages.
Consequence of unauthorized use	Unauthorized use of the PAK allows a malicious party to create forged key stream messages. Possibly causing denial of service attacks, or allowing access to attack vectors that would not have been available otherwise.
Mitigation	The rights layer implementation should be robust, and it should not reveal the PAK to the key stream layer (or any other party). It should only provide a “authentic / not authentic” response for key stream messages.
Asset	DEK

Function	The Device Entitlement Key. This key is used to encrypt and decrypt SEKs or PEKs in BCROs. This key is derived either from the BGKs, the UGK or the UGK and the BCI field from the BCRO. A DEK is specific to a single BCI-value and broadcast sub-group in the non-interactive mode.
Consequence of unauthorized use	Unauthorized use of a DEK allows unauthorized use of a BCRO. A DEK is additionally required to compute a BAK, but is not alone sufficient for that. However, since both the RIAK and BGKs, UGK and UDK reside in the registration layer, one must assume that unauthorized use of a DEK could also allow unauthorized use of a RIAK.
Mitigation	The DEK should be provided by the registration layer to the rights layer, and neither party should reveal it to unauthorized elements in the system. The DEK can also be changed relatively frequently if necessary by changing the BCI in BCROs. This allows a method to recover from the compromise of a single DEK.
Asset	IPsec SA
Function	The IPsec SA defines the decryption and encryption of traffic at the traffic layer if IPsec is used. The IPsec SA is instantiated by the Key Stream Layer.
Consequence of unauthorized use	Encrypted plaintext could be decrypted without authorization. A device could send forged packets to other devices. This could lead to a denial-of-service attack or access to attack vectors that would otherwise have been unavailable.
Mitigation	An IPsec implementation should not export or reveal the cryptographic keys related to IP datacast to unauthorized parties. Additionally the IPsec implementation should not allow the redirection of data decrypted using IP datacast SAs to non-robust or compliant applications. Especially if content protection is desired.
Asset	SRTP Crypto Context
Function	The SRTP crypto context defines the decryption and encryption of traffic at the traffic layer if SRTP is used. The SRTP Crypto Context is instantiated by the Key Stream Layer.
Consequence of unauthorized use	Encrypted plaintext could be decrypted without authorization. A device could send forged packets to other devices. This could lead to a denial-of-service attack or access to attack vectors that would otherwise have been unavailable.
Mitigation	An SRTP implementation should not export or reveal the cryptographic keys related to IP datacast to unauthorized parties. Additionally the SRTP implementation should not allow the redirection of data decrypted for IP datacast to non-robust or compliant applications. Especially if content protection is desired.

History

Document history		
	10-Mar-05	1 st draft release to DVB-CBMS
	7-Jun-05	Release to DVB-CBMS