DRM Specification - SCE Extensions

Approved Version 1.0 – 05 Jul 2011

Open Mobile Alliance

OMA-TS-SCE_DRM-V1_0- 20110705-A

Contents

Figures

Tables

1. Scope

Open Mobile Alliance (OMA) specifications are the result of continuous work to define industry-wide interoperable mechanisms for developing applications and services that are deployed over wireless communication networks.

The scope of OMA "Digital Rights Management" (DRM) is to enable the distribution and consumption of digital content in a controlled manner. The content is distributed and consumed on authenticated Devices per the usage rights expressed by the content owners. OMA DRM work addresses the various technical aspects of this system by providing appropriate specifications for content formats, protocols, and a rights expression language.

A number of DRM specifications have already been defined within the OMA. The latest approved release of the OMA DRM enabler ([DRM-v2.1], including [DRM-DRM-v2.1], [DRM-DCF-v2.1], [DRM-REL-v2.1]), is referred to within this document as "OMA DRM 2.1".

This specification defines the ROAP protocol suite extensions necessary to implement necessary functions, as required per [SCE-RD]. More specifically, this specification will specify the interface SCE-1-ROAP as defined in [SCE-AD].

2. References

2.1 Normative References

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. November 1997, URL:http://www.ietf.org/rfc/rfc2234.txt

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

[DRM-v2.1] The OMA DRM 2.1 enabler as described in "Enabler Release Definition for DRM V2.1, Approved Version 2.1", OMA-ERELD-DRM-V2_1-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[DRM-DRM-v2.1] "DRM Specification, Draft Version 2.1", OMA-TS-DRM-DRM-V2_1-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[DRM-REL-v2.1] "DRM Rights Expression Language, Draft Version 2.1", OMA-TS-DRM-REL-V2_1-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[DRM-DCF-v2.1] "DRM Content Format, Draft Version 2.1", OMA-TS-DRM-DCF-V2_1-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[SCE-RD] "Secure Content Exchange Requirements, Draft Version 1.0", OMA-RD-SCE-V1_0-20060908-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[SCE-AD] "Secure Content Exchange Architecture, Draft Version", OMA-AD-SCE-V1_0-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[SCE-GEN] "SCE Generic Mechnisms, Draft Version"

OMA-TS-SCE_GEN-V1_0-D, Open Mobile Alliance™,

URL:http://www.openmobilealliance.org/

[SCE-DRM] "DRM Specification – SCE Extensions, Draft Version", OMA-TS-SCE_DRM-V1_0-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[SCE-REL] "DRM Rights Expression Language – SCE Extensions, Draft Version", OMA-TS-SCE_REL-V1_0-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[SCE-LRM] "DRM Local Rights Management, Draft Version", OMA-TS-SCE_LRM-V1_0-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[SCE-DOM] "DRM User Domains, Draft Version", OMA-TS-SCE_DOM-V1_0-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[SCE-A2A] "DRM Agent-to-Agent transfer, Draft Version", OMA-TS-SCE_A2A-V1_0-D, Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[PKCS-1] "PKCS #1 v2.1: RSA Cryptography Standard", RSA Laboratories. June 2002. http://www.rsasecurity.com/rsalabs

[SHA-1] NIST FIPS 180-2: Secure Hash Standard. August 2002.

http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[XML-DSIG] XML-Signature Syntax and Processing. D. Eastlake, J. Reagle, and D. Solo. W3C Recommendation, February 2002. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

[XML-Enc] XML Encryption Syntax and Processing. D. Eastlake and J. Reagle. W3C Recommendation, December 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[XML-Schema] XML Schema Part 1: Structures D. Beech, M. Maloney, and N. Mendelsohn. W3C Recommendation, May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

XML Schema Part 2: Datatypes. P. Biron and A. Malhotra. W3C Recommendation, May 2001.

http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[OCSP] Myers, M., Ankney, R., Malpani, A., Galperin, S. and C. Adams, "Internet X.509 Public Key Infrastructure: Online Certificate Status Protocol - OCSP", RFC 2560, June 1999. http://www.ietf.org/rfc/rfc2560.txt

[OCSP-MP] OMA Online Certificate Status Protocol (profile of [OCSP]) V 1.0, http://www.openmobilealliance.org/

[ODRL] “Open Digital Rights Language (ODRL)”, Version 1.1, 8 August 2002, URL:http://odrl.net/1.1/ODRL-11.pdf or URL:http://www.w3.org/TR/odrl/

2.2 Informative References

[OMADICT] “Dictionary for OMA Specifications”, Version x.y, Open Mobile Alliance™, OMA-ORG-Dictionary-Vx_y, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Constraint	A restriction on a Permission over DRM Content (DRM V2.0).
Consume	To Play, Display, Print or Execute DRM Content on a Device or to render DRM Content on a Render Client.
Content	One or more Media Objects (DRM V2.0).
Content Issuer	The entity making content available to the DRM Agent in a Device (DRM V2.0).
Content Provider	An entity that is either a Content Issuer or a Rights Issuer (DRM V2.0).
Device	A Device is the entity (hardware/software or combination thereof) within a user equipment that implements a DRM Agent. The Device is also conformant to the OMA DRM specifications. The Device may include a smart card module (e.g. a SIM) (DRM V2.0).
Device Rights Object (Device RO)	A Rights Object that is initially targeted to a specific entity. Subsequently, the Rights Object may be allowed to be targeted to other entities to be consumed, serially or in parallel, independently of membership in a Domain or User Domain.
Domain	A set of v2.x and/or SCE DRM Agents that can consume Domain Rights Objects.
Domain Rights Object (Domain RO)	A Rights Object that is targeted to a specific v2.x Domain. The Rights Object can be consumed independently by each v2.x or SCE DRM Agent that is a member of the Domain.
DRM Agent	The entity in the Device that manages Permissions for Media Objects on the Device (DRM V2.0).
DRM Content	Media Objects that are consumed according to a set of Permissions in a Rights Object (DRM V2.0).
DRM Time	A secure, non user-changeable time source. The DRM Time is measured in the UTC time scale (DRM V2.0).
Local Rights Manager (LRM)	An entity that is responsible for aspect(s) of Import and it may also manage an Imported-Content for a limited group of OMA DRM Agents.
Move	To make Rights existing initially on a source Device fully or partially available for use by a recipient Device, such that the Rights or parts thereof that become usable on the recipient Device can no longer be used on the source Device.
Rights	The collection of permissions and constraints defining under which circumstances access is granted to DRM Content.
Permission	Actual usages or activities allowed (by a Rights Issuer or Local Rights Manager) over DRM Content.
Source Device	The Device that sends out Rights.
Recipient Device	The Device that is final destination which receives Rights.
State Information	A set of values representing current state associated with Rights. It is managed by the DRM Agent only when the Rights contain any of the stateful constraints (e.g. interval, count, timed-count, accumulated, etc.).

Superdistribution	A mechanism that (1) allows a User to distribute DRM Content to other Devices through potentially insecure channels and (2) enables the User of that Device to obtain a Rights Object for the superdistributed DRM Content (DRM V2.0).
User	The human user of a Device. The User does not necessarily own the Device (DRM V2.0).
User Domain	A set of v2.x and/or SCE DRM Agents that can consume User Domain Rights Objects.
User Domain Rights Object (User Domain RO)	A Rights Object that is targeted to a specific User Domain. Besides requiring membership in the User Domain, consumption may require being targeted to an SCE DRM Agent.

3.3 Abbreviations

OMA	Open Mobile Alliance
ROAP	Rights Object Acquisition Protocol
RI	Rights Issuer
CI	Content Issuer
RO	Rights Object
CEK	Content Encryption Key
REK	Rights Object Encryption Key
LRM	Local Rights Manager
DCF	DRM Content Format
DER	Distinguished Encoding Rules
HTTP	HyperText Transfer Protocol
KEK	Key Encryption Key
MAC	Message Authentication Code
OCSP	Online Certificate Status Protocol
ODRL	Open Digital Rights Language
PDU	Protocol Data Unit
SCE	Secure Content Exchange
URL	Uniform Resource Locator
XML	Extensible Markup Language

4. Introduction

This document specifies the SCE-1-ROAP interface that is defined in [SCE-AD]. Since the SCE-1-ROAP interface is defined as an extension version of ROAP 1.0 [DRM-DRM-v2.1], all protocols defined in OMA DRM 2.1 are supported, but also new protocols are defined to support the following functions:

- Acquire a User Domain RO from either an RI or an LRM
- Move Device RO via an RI
- Move <userDomain>-constrained RO via an RI
- Rights Object upgrade

This document describes the new ROAP Triggers, syntax and semantics of the new messages, and the protocol processing rules for the new functions.

5. ROAP Suite Overview

This technical specification extends ROAP Suite from [DRM-DRM-v2.1].

5.1 Device RO Acquisition

SCE Devices can acquire Device ROs from either an RI or an LRM (with the **`oma-kp-localRightsManagerDevice`** key purpose) using ROAP.

5.2 Domain RO Acquisition

SCE Devices can acquire Domain ROs only from an RI. An LRM MUST NOT deliver Domain ROs to an SCE Device.

5.3 User Domain RO Acquisition

SCE Devices and v2.x Devices can acquire User Domain ROs. An SCE Device is distinguished from a v2.x Device by the inclusion of the **`oma-kp-sceDrmAgent`** key purpose in the Device's certificate [SCE-A2A].

5.3.1 By SCE Devices

SCE Devices can acquire User Domain ROs from either an RI or LRM (with the **`oma-kp-localRightsManagerDomain`** key purpose). The acquisition can be by any mechanism allowed by [DRM-DRM-v2.1]. However, if the User Domain RO has the <userDomain> constraint ([SCE-REL]), then the User Domain RO MUST be acquired by the Device initially targeted by the RI or LRM via ROAP and the extensions that are defined in section 8.1. An SCE Device can be targeted by another SCE Device for delivery of a <userDomain>-constrained RO either via an A2A protocol transaction or operation [SCE-A2A] or, via the Move <userDomain>-constrained RO via RI protocol defined in section 8.3. In the latter case, the RI delivers the resulting <userDomain>-constrained RO using a ROAP RO response with the *UserDomainConstrainedROMoved* extension defined in section 8.1(where the recipient SCE Device need not be a member of the User Domain in order to acquire the RO).

User Domain ROs without the <userDomain> constraint can be acquired by an SCE Device, whether or not the Device is a member of the User Domain.

5.3.2 By v2.x Devices

V2.x Devices can acquire User Domain ROs from either an RI or LRM under the following conditions:

- The RI or LRM supports the proxy join User Domain mechanism described in [SCE-DOM]. Note that the proxy join User Domain mechanism MAY be disabled by either the DA or DEA for a particular User Domain.
- If the RO is acquired from an LRM, then the LRM MUST have the **`oma-kp-rightsIssuer`** key purpose.
- The User Domain RO is fully compatible with a v2.x Domain RO. In particular, the RO MUST NOT have the <userDomain> constraint.

If these conditions are met, then the RI or LRM can deliver the User Domain RO to a v2.x Device using any mechanism allowed by [DRM-DRM-v2.1].

5.4 The 2-pass Move RO Protocol

5.4.1 The 2-pass Move Device RO Protocol

The 2-pass Move Device RO protocol is the protocol by which a Source Device transfers one or more ROs originally issued by an RI or LRM to an RI, and the transferred ROs are intended to be Moved to a Recipient Device. The Source Device MUST be an SCE Device. The Recipient Device MAY be an SCE Device. The Recipient Device MAY alternatively be a v2.x Device.

If the original Issuer of the RO is an RI, the RI as the responder of this protocol MUST be that original Issuer RI. If the RO is originally issued by an LRM, the LRM-created RO MUST contain information specifying which RI(s) it designates to Move the RO (i.e.<moveIndication> element, see [SCE-REL]). An RI SHOULD verify that it has agreed to provide Move service for that LRM in accordance with the result of the LRM-RI registration protocol [SCE-LRM]. If the original Issuer of the RO is an LRM, once the RO is Moved via an RI, the RI as future responder of this protocol MUST be that same RI. This protocol assumes that the Source Device already has a valid RI context for the RI that it requests to Move the RO.

This protocol includes mutual authentication between Device and RI, integrity-protected request and response, and transferring of ROs. This protocol further includes the verification of trust relationship between the RI and the LRM in case that the RO was originally issued by an LRM. This protocol ensures that the RI is able to verify the RI-generated or LRM-generated signature over the RO, so as to determine that the received RO was last issued by the same RI or as originally issued by an LRM trusted by the RI. This protocol MAY involve OCSP protocol between RI and OCSP Responder for checking status of RI's certificate chain. After successful 2-pass Move Device RO Protocol execution, the RI MUST conduct RO Acquisition Protocol including optional ROAP-ROAcquisition Trigger as per section 8.1, with the Recipient Device.

NOTE: Although the RO Acquisition Protocol itself is not part of the 2-pass Move Device RO protocol, the RI SHOULD NOT include the <signature> element over the <rights> element in the resultant RO if the RO is intended for delivery to a v2.x Device, where a v2.x Device is distinguishable from an SCE-conformant Device by the absence of the **oma-kp-sceDrmAgent** key purpose in its certificate (see [SCE-A2A]).

.

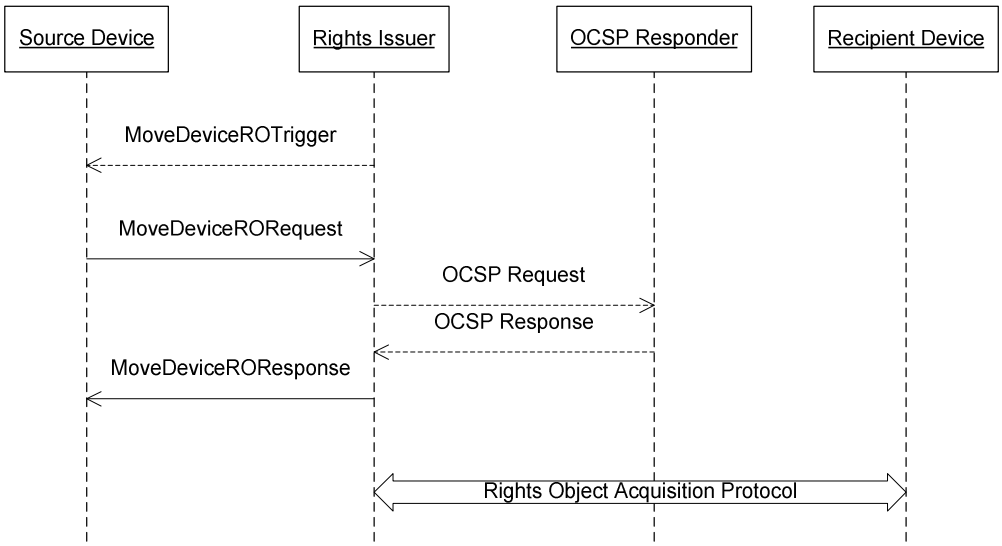

Figure 1 the 2-pass Move Device RO Protocol

5.4.2 The 2 pass Move <userDomain>-constrained RO Protocol

The 2-pass <userDomain>-constrained RO Protocol is the protocol by which a Source Device enlists the services of an RI to Move one or more Rights (User Domain ROs that have the <userDomain> constraint) issued by an RI or LRM to a designated Recipient Device. The Source Device and the Recipient Device MUST be a members of the User Domain(s) the ROs are bound to; and the RI MUST be authorized to manage the User Domain(s) by the related DEA(s). This protocol assumes that the Source Device and the Recipient Device each have a valid RI context for the associated RI.

This protocol includes mutual authentication between the Source Device and the RI, integrity-protected request and response, and transfer of Rights. This protocol ensures that REK for each RO being Moved is not exposed to the RI. This protocol MAY involve OCSP protocol between the RI and an OCSP Responder for checking status of RI's certificate chain. After

successful 2-pass Move <userDomain>-constrained RO Protocol execution, the RI MUST conduct RO Acquisition Protocol including optional ROAP-ROAcquisition Trigger as per section 8.1, with the Recipient Device. But the RO Acquisition Protocol itself is not part of this protocol. When creating RO for the Recipient Device, rather than generating a new REK, the RI uses the key it received from the Source Device.

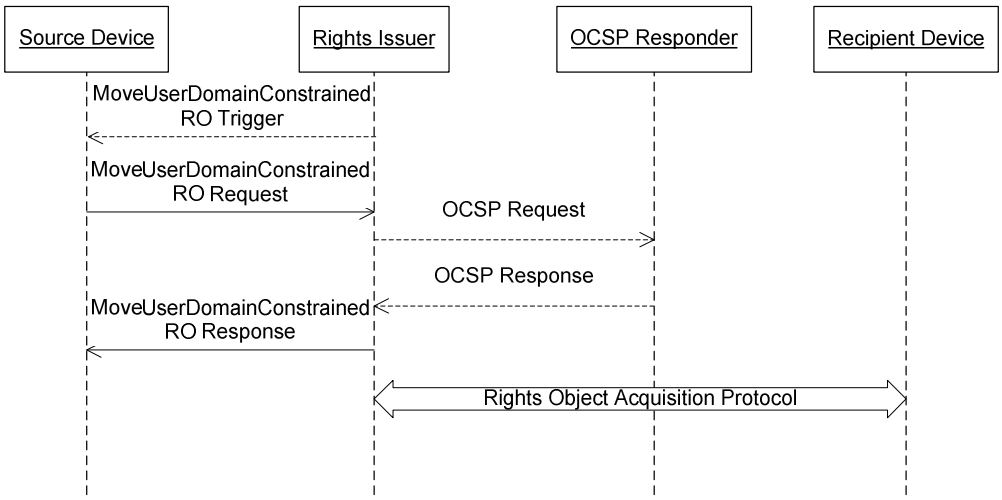

Figure 2 the 2-pass Move <userDomain>-constrained RO Protocol

5.5 The 2-pass RO Upgrade Protocol

The 2-pass RO Upgrade protocol is the protocol by which a Device upgrades an existing RO. This protocol is intended for adding permissions to existing Rights managed by the DRM Agent, in the case where the existing Rights don't contain the required permissions. The added permission can be, for example, for sharing (e.g. Move or Copy). This results in a new Rights Object that replaces the old Rights Object. This protocol includes mutual authentication of Device and RI, integrity-protected request and the secure transfer of the existing RO (and corresponding State Information if it is stateful), User desired permission and new RO which is intended to substitute the existing RO. This protocol ensures the RI is able to verify the existing RO coming from the Device was originally issued by itself. This protocol MAY involve OCSP protocol between RI and OCSP Responder for checking status of RI's certificate chain. The successful execution of this protocol assumes the Device to have a pre-established RI Context with the RI.

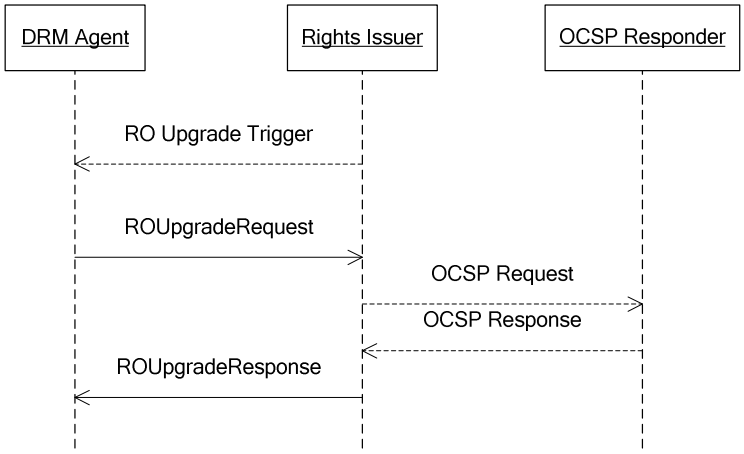

Figure 3 the 2-pass RO Upgrade Protocol

5.6 XML Namespace

In this version and minor upgrade of the specification, the XML namespace URI must be "*urn:oma:xml:sce:roap*". For the sake of convenience, this specification uses the namespace prefix as "sceroap", the namespace prefix for OMA DRM 2.1 ROAP schema as "roap" and the namespace prefix for generic schema for SCE as "gen", the namespace prefix for [XML-DSIG] as "ds" and the namespace prefix for [XML-Enc] as "xenc" (see [SCE-GEN]).

6. ROAP Initiation

ROAP and SCE-ROAP MAY be initiated by a ROAP Trigger. In addition to the triggers defined in [DRM-DRM-v2.1], SCE defines the following new triggers:

- ROAP-MoveDeviceRO Trigger
- ROAP-MoveUserDomainConstrainedRO Trigger
- ROAP-ROUpgrade Trigger

These new triggers are based on the trigger defined in [SCE-GEN]. The schema fragments shown in this section represent the elements that should be present under the <gen:trgInfo> element.

For the information of <gen:trgInfo> element, refer to [SCE-GEN].

The DRM Agents that conform to this version of the specification MUST support ROAP triggers which are defined in [DRM-DRM-v2.1].

6.1 MoveDeviceRO Trigger

A ROAP-MoveDeviceROTrigger MAY be delivered from an RI to a DRM Agent to initiate the ROAP-MoveDeviceRO protocol. The root element of the message MUST be a <gen:drmTrigger> element as specified in [SCE-GEN], in which the following elements are present:

element / attribute	usage	value
id	O	Default, as specified in [SCE-GEN]
type	M	“moveDeviceRO”
version	M	“1.0”
resID	M	RI ID
resAlias	O	Default, as specified in [SCE-GEN]
nonce	O	Default, as specified in [SCE-GEN]
reqURL	M	Default, as specified in [SCE-GEN]
body	M	Specified below

Table 1 MoveDeviceRO Trigger Message Parameters

The MoveDeviceRO Trigger contains a <gen:body> element that MUST have a <gen:trgInfo> child element which MUST have a <sceroap:moveDeviceROTrgInformation> element as defined by the following XML schema fragment:

```
<element name="moveDeviceROTrgInformation">
  <complexType>
    <sequence>
      <element name="recipientInfo" type="string" minOccurs="0" />
    </sequence>
    <attribute name="roRequested" type="boolean" default="true" />
  </complexType>
</element>
```

If the User of Source Device has designated the Recipient Device in the RI portal, the RI MUST add a <recipientInfo> element in the <sceroap:moveDeviceROTrgInformation> element. The value of <recipientInfo> element identifies information of Recipient Device and SHOULD be a string which is identifiable by user and it can be e.g. phone number, user name.

Depending on the RI policy, the RI may record Rights Objects while those were issued. In such case, the RI does not have to retrieve all information about the Rights Object. Hence, if the RI already recorded the issued Rights Object, the RI MUST set the roRequested attribute to 'false' in the <sceroap:moveDeviceROTrgInformation> element. If the RI did not record the Rights Object, the RI MUST set the roRequested attribute to 'true' or omit the roRequested attribute in the <sceroap:moveDeviceROTrgInformation> element.

In the case that a DRM Agent receives a MoveDeviceROTrigger, the DRM Agent SHALL obtain user's consent and then initiate the ROAP-MoveDeviceRO protocol, with the following exceptions. If the DRM Agent has an invalid RI Context for the specified <riID> in the trigger, the DRM Agent MUST initiate the 4-pass ROAP-Registration protocol by using <roapURL> element in the trigger.

6.2 MoveUserDomainConstrainedRO Trigger

A ROAP-MoveUserDomainConstrainedRO Trigger MAY be delivered from an RI to a DRM Agent to initiate the ROAP Move User Domain Constrained RO protocol. The root element of the message MUST be a <gen:drmTrigger> element as specified in [SCE-GEN], in which the following elements are present:

element / attribute	usage	value
id	O	Default, as specified in [SCE-GEN]
type	M	"moveUserDomainConstrained"
version	M	"1.0"
resID	M	RI ID
resAlias	O	Default, as specified in [SCE-GEN]
nonce	O	Default, as specified in [SCE-GEN]
reqURL	M	Default, as specified in [SCE-GEN]
body	M	Specified below

Table 2 MoveUserDomainConstrainedRO Trigger Message Parameters

The MoveUserDomainConstrainedRO Trigger contains a <gen:body> element that MUST have a <gen:trgInfo> child element which MUST have a <sceroap:moveUserDomainConstrainedROTrgInformation> element as defined by the following XML schema fragment:

```
<element name="moveUserDomainConstrainedROTrgInformation">
  <complexType>
    <sequence>
      <element name="recipientInfo" type="string" minOccurs="0" />
    </sequence>
  </complexType>
</element>
```

In the case that a DRM Agent receives a ROAP-MoveUserDomainConstrainedRO Trigger, the DRM Agent SHALL obtain user's consent and then initiate the ROAP-MoveUserDomainConstrainedRO protocol, with the following exceptions. If the DRM Agent has an invalid RI Context for the specified <riID> in the trigger, the DRM Agent MUST initiate the 4-pass ROAP-Registration protocol by using <roapURL> element in the trigger.

6.3 ROUpgradeTrigger

An ROAP-ROUpgradeTrigger MAY be delivered from an RI to a DRM Agent to initiate the ROAP ROUpgrade protocol. The root element of the message MUST be a <gen:drmTrigger> element as specified in [SCE-GEN], in which the following elements are present:

element / attribute	usage	value
id	O	Default, as specified in [SCE-GEN]
type	M	“roUpgrade”
version	M	“1.0”
resID	M	RI ID
resAlias	O	Default, as specified in [SCE-GEN]
nonce	O	Default, as specified in [SCE-GEN]
reqURL	M	Default, as specified in [SCE-GEN]
body	M	Specified below

Table 3 ROUpgrade Trigger Message Parameters

The ROUpgrade Trigger contains a <gen:body> element that MUST have a <gen:trgInfo> child element which MUST have a <sceroap:roUpgradeTrgInformation> element as defined by the following XML schema fragment:

```
<element name="roUpgradeTrgInformation">
  <complexType>
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element name="roID" type="ID" minOccurs="0" />
      <element name="roAlias" type="string" minOccurs="0" />
      <element name="upgradeInfo" type="string" minOccurs="0" />
    </sequence>
    <attribute name="roRequested" type="boolean" default="true" />
  </complexType>
</element>
```

The <sceroap:roUpgradeTrgInformation> element MAY carry *roID*, *roAlias* and *upgradeInfo*. The *roID* and *roAlias* specify the RO to be upgraded. The *upgradeInfo* is textual information for the user to confirm his/her request made in browse session about upgrading an RO, e.g. what existing RO is, what additional permission is.

When an ROUpgrade trigger is received, the DRM Agent SHALL show the user the *upgradeInfo* (if it is present) and obtain the user’s consent on whether or not to initiate 2-pass RO Upgrade protocol.

If the RO Upgrade protocol is initiated at that occasion, the ROAP-ROUpgradeRequest SHALL be constructed according to the information carried in the <sceroap:roUpgradeTrgInformation> element (for detail, see section 6.3).

6.4 Initiating ROAP from a BatchRIURL for multiple DCFs

This section provides the efficient mechanism for a device to acquire multiple ROs at one time execution of the ROAP instead of multiple execution of the ROAP. This is particularly useful for a device to acquire multiple ROs when devices exchange multiple contents via wired communication, wireless communication or a removable media.

This section applies ONLY to Connected Devices.

If the DRM Agent receives multiple DCFs with the same BatchRIURL header, the DRM Agent MAY attempt to acquire Rights for multiple DCFs with the same BatchRIURL header with a specified BatchRIURL, by sending a HTTP request message to the BatchRIURL stored in multiple DCFs and responding to the ROAP Trigger that will be returned by the RI.

6.4.1 BatchRIURL header

The *BatchRIURL* header is an indication to the client that multiple Rights Objects for DRM Contents can be obtained from the RI.

```
BatchRIURL = “BatchRIURL” “:” batchri-url
batchri-url = token
```

The parameter `batchri-url` MUST be a URL according to [RFC2396] and a successful request to the URL MUST return a ROAP Trigger as defined in [DRM-DRM-v2.1]. If `batchri-url` is a HTTP URL and the request fails with error code `404 Not Found` [RFC2616], the Device SHOULD NOT make further requests to the URL. If the request fails with some other error, the Device MAY retry the request at a later time.

7. New ROAP Types

This section defines the new ROAP types defined in this specification

7.1 The RightsInfo type

The element of sceroap:RightsInfo type carries information of the Rights to be transferred to the RI in the case of ROAP-MoveDeviceRORequest and ROAP-ROUpgradeRequest. The schema of sceroap:RightsInfo type is defined as follows:

```
<complexType name="RightsInfo">
  <sequence>
    <element name="roID" type="ID" />
    <element name="rights" type="o-ex:rightsType" minOccurs="0" />
    <element name="signature" type="ds:SignatureType" minOccurs="0" />
    <element name="stateInfo" type="o-ex:constraintType" minOccurs="0" />
    <element name="sourceDeviceID" type="gen:Identifier" />
    <element name="encKey" type="xenc:EncryptedKeyType" />
    <element name="mac" type="base64Binary" />
  </sequence>
</complexType>
```

As indicated by the XML schema, an element of sceroap:RightsInfo type includes one mandatory <roID> element, <encKey>, one optional <rights> element, one optional <signature> element, zero or more <stateInfo> elements, one mandatory <sourceDeviceID> element, one mandatory <encKey> element, and one mandatory <mac> element. These elements are specified as follows:

<roID> carries the ID of the Rights that will be transferred to the RI, so its value MUST be the same as the ID of the original RO.

<rights> is the <rights> element in the original RO. In the case of ROAP-MoveDeviceRORequest and ROAP-ROUpgradeRequest, if the corresponding trigger (ROAP-MoveDeviceRO Trigger or ROAP-ROUpgrade Trigger) indicates (by the 'roRequested' attribute) that the RI does not keep record of the issued ROs, this element SHALL be present; else this element SHALL not be present.

<signature> carries the signature of the RI or LRM over the <rights> element. If the <rights> element is present, then this element SHALL also be present; else this element SHALL not be present.

If the Rights to be transferred are stateful, the <stateInfo> element MUST be present and MUST appear at least once. The <stateInfo> element carries the data maintained for stateful <constraint> elements inside the <rights> element.

<sourceDeviceID> carries the ID of the Device that sends the request message to the RI (i.e. the SHA-1 hash of the DER-encoded subjectPublicKeyInfo value in its certificate).

<encKey> contains a wrapped concatenation of a MAC key, K_{MAC} and a RO Encryption Key, K_{REK} (see section 13.1.1 for details). The Id attribute of this element SHALL be present and SHALL have the same value as the value of the URI attribute of the <ds:RetrievalMethod> element in the <ds:KeyInfo> element inside the <rights> element. The <ds:KeyInfo> child element of the <encKey> element SHALL identify the wrapping key. The <ds:KeyInfo> element SHALL be the <gen:X509SPKIHash> element, identifying the RI Public Key through the (SHA-1) hash of the DER-encoded subjectPublicKeyInfo value in the RI certificate

The <mac> element provides integrity protection through a MAC on the canonical version conforming to [DRM-DRM-v2.1] of the element of sceroap:RightsInfo type (excluding the <mac> elements) using the MAC key, K_{MAC} wrapped in the <encKey> element. The MAC algorithm SHALL be the same algorithm that was negotiated as part of the registration with the RI i.e. the MAC algorithm stored in the RI Context.

7.1.1 Processing Rules from DRM Agent side

- In order to package an element of sceroap:RightsInfo type**,** the DRM Agent MUST take the following into account:

- If 'roRequested' attribute in the ROAP-MoveDeviceRO Trigger or ROAP-ROUpgrade Trigger indicates that the RI does not keep record of the issued ROs, or the user initiates the protocol, then the DRM Agent MUST include the <rights> and <signature> elements (which MUST be identical to the element stored at the installation time). Else, the DRM Agent SHALL NOT include the <rights> and <signature> element.
- The <encKey> element MUST be set to the concatenation of K_{REK} (from the original RO) and K_{MAC} (which is a randomly generated 128-bit long MAC Key) after being wrapped using the RI's Public Key (see section 13.1.1).
- The <mac> element MUST be set to the MAC calculated on the canonical version of the <rightsInfo> element of sceroap:RightsInfo type (excluding the <mac> element) using the K_{MAC} and the MAC algorithm from the RI Context.

7.1.2 Processing Rules from RI side

In order to verify an element of sceroap:RightsInfo type, the RI MUST perform the following checks:

1. Check that the signature of the request message (e.g. ROAP-MoveDeviceRORequest or ROAP-ROUpgradeRequest) is generated by the Device with the same ID as <sourceDeviceID> element. If check fails, the RI sends a response message containing error status *InvalidRO*.

2. In case that the RI keeps records of <rights> and <signature> elements, unless there is a <moveIndication> element in the request, check that the ROID in the request message can be found in the RI's issue history. If check fails, the RI sends a response message containing error status *NotFound*.

3. If the request is of type Move Device RO and this RI tracks the number of times that it Moves Device ROs that are all derived from a single initial Device RO that it originated or that an LRM originated (possibly in order to limit the number of such Moves), then check for any prior records. (Note that if A2A Move is permitted by the original RO, the RI can not track all Moves relating to the original RO (whether or not the RI or an LRM issued the original RO).)

4. If the current request includes a <moveIndication> element, then the RI sends a response message containing error status *InvalidRO* if this RI's ID is not included in the <moveIndication> element or if for some reason it does not trust all of the other RIs included within the <moveIndication> element (since if the 'allowPartial' attribute equals "true" then multiple RIs identified by the <moveIndication> element can legitimately Move Device ROs tracing back to the original Device RO (where each such RI gains access to the REK), and if the 'allowPartial' attribute equals "false" then a rogue Device can nevertheless Move the Device RO via multiple such RIs).

5. If the <rights> element is present, check its sibling <signature> elements is also present. If check fails, the RO sends a response message containing error status InvalidRO.

6. Verify the legitimacy of the <signature> element of step 5 as follows:

   - Identify the signer of signature by looking <ds:KeyInfo> child element. If there is error during identifying signer, the RI sends a response message containing appropriate error. (i.e. *NoCertificateChain, InvalidCertificateChain or TrustedRootCertificateNotPresent*. If the signer is neither the RI as a recipient of the request message nor any LRM that has registered to this RI with a request for RO Move service (see [SCE-LRM]), the RI sends a response message containing error status *UnknownRO*.

   - Validate signature value. If the signature validation fails, the RI sends a response message containing error status *SignatureError*.

7. If any <stateInfo> element is present, verify for all <stateInfo> elements that the state information is consistent with the original stateful <constraint> elements in the <rights> element. If verification fails, the RI sends a response message containing error status *InvalidRO*.

8. Decrypt K_{REK} and verify $K_{MAC,}$ as follows:

   - Unwrap K_{REK} and K_{MAC} (see section 13.1.1).

- Calculate a MAC on the canonical version of the element of sceroap:RightsInfo type (excluding the <mac> element) using the K_{MAC}. The MAC algorithm to use is defined in the Device Context.
- Check the calculated value against the <mac> element of the element of sceroap:RightsInfo type . If the calculated value is not equal to value of the <mac> element, the RI MUST send a response message with error status *invalidRO*.

9. Do AES-UNWRAP of Content Encryption Key (CEK) using the decrypted K_{REK}. If any error occurred during AES-UNWRAP of CEK, the RI regards that the requesting DRM Agent did not package the K_{REK} properly and sends an response message containing error status *InvalidRO*.

7.2 State information

The <stateInfo> element is of type o-ex:constraintType [ODRL] and MUST be repeated for every stateful constraint in the original <ro> element that contains an ”id” attribute. A stateful constraint is a <constraint> element that contains one of the following elements: <count>, <timed-count>, <interval> or <accumulated>.

In case of RO Upgrade, <stateInfo> element carries the remaining Rights of the specific RO to be upgraded. For the <count> and <timed-count> elements, the value contains the remaining count value. For the <accumulated> element, the value contains the remaining duration that the Content can be rendered (in the format of the <accumulated> element). The <interval> element is handled differently. If the Content has not been rendered, i.e. the interval has not started, then nothing is placed in the <stateInfo> element. If the Content has been rendered, i.e. the interval has been started, then the <interval> element is transformed into a <datetime><end>xx</end></datetime>, where xx is the end date/time after which the Content can not be rendered.

In case of Move, <stateInfo> element carries the Rights to be Moved to the designated Recipient Device. For the <count> and <timed-count> elements, the value contains the count the Recipient Device can use, which must be equal to or less than the current remaining count value on the Source Device. No other splitting of the RO is allowed.

8. New ROAP Messages

These sections define the additions to v2.x ROAP messages. When a new message is based on a message in [SCE-GEN], then the Requester of the protocol is a Device and the responder of the protocol is an RI or an LRM. Hence the value of <reqID> element must be an identifier of the Device and the value of <resID> element must be an identifier of the RI or LRM.

8.1 RO Acquisition

This specification does not re-define XML Schema for RO Acquisition protocol. The DRM Agent must use ROAP schema defined in [DRM-DRM-v2.1] for RO Acquisition protocol. A ROAP-ROAcquisition Trigger MAY be delivered from an RI or LRM to a DRM Agent to initiate the RO Acquisition protocol.

8.1.1 RO Request

The ROAP-RORequest message is sent from a Device to an RI or LRM to request Rights Objects. This message is the first message of the 2-pass RO Acquisition protocol.

ROAP-RORequest	
Parameter	**Mandatory/Optional**
Device ID	M
Domain ID	O
RI ID	M
Device Nonce	M
Request Time	M
RO Info	M
Certificate Chain	O
Extensions	O
Signature	M

Table 4 RO Request Message Parameters

Device ID identifies the requesting Device. The value MUST equal the stored Device ID.

Domain ID, when present, identifies the Domain or User Domain for which the requested ROs shall be issued.

RI ID identifies the authorizing RI or LRM. The value MUST equal the stored RI ID or LRM ID.

Device Nonce is a nonce chosen by the Device. Nonces are generated and used in this message.

Request Time is the current DRM Time, as seen by the Device.

RO Info identifies the requested Rights Object(s). The parameter consists of a (non-empty) set of Rights Object identifiers identifying the requested Rights Objects, and for each RO identifier an optional hash of the DCF associated with the requested RO. The DCF hash SHOULD be included when the Device is in possession of the associated DCF, unless its inclusion, as determined by some vendor-specific algorithm, would be impractical (e.g. due to the size of the DCF). If the 2-pass protocol is initiated by a ROAP Trigger, the Device SHOULD use the <contentID> elements of the ROAP Trigger to identify the associated DCF(s) over which a DCF hash should be calculated. The DCF hash, if computed, MUST be computed as specified in [DRM-DCF-v2.1] using the SHA-1 algorithm.

In case the RO refers to many OMA DRM container (i.e. CID), the Agent MAY send multiple DCF Hashes (one per file containing a DRM container referred by the RO) by duplicating the <roID> in the sequence. Refer to Annex G.1.6 in [DRM-DRM-v2.1] for an example of multiple DCF Hashes case.

Certificate Chain: This parameter is sent unless it is indicated in the RI or LRM Context that this RI or LRM has stored necessary Device certificate information. When present, the parameter value SHALL be as described for the Certificate Chain parameter in the ROAP-RegistrationRequest message.

Extensions: The following extensions are defined for the ROAP-RORequest message:

Peer Key Identifier: An identifier for an RI or LRM public key stored in the Device. If the identifier matches the stored RI ID or LRM ID, it means the Device has already stored the RI ID or LRM ID and the corresponding RI or LRM certificate chain, and the RI or LRM does not need to include its certificate chain in its response message. The Device MUST send the Peer Key Identifier extension if, and only if, it has stored the RI or LRM public key corresponding to the stored RI ID or LRM ID.

No OCSP Response: Presence of this extension indicates to the RI or LRM that there is no need to send any OCSP responses since the Device has cached a complete set of valid OCSP responses for this RI or LRM. The Device MUST send the No OCSP Response extension if, and only if, it has a complete set of valid OCSP responses for the RI or LRM certificate chain.

OCSP Responder Key Identifier: This extension identifies an OCSP responder key stored in the Device. If the identifier matches the key in the certificate used by the RI's or LRM's OCSP responder, the RI or LRM MAY remove the OCSP Responder certificate chain from the OCSP response before providing the OCSP response to the Device. The Device MUST send the OCSP Responder Key Identifier extension if, and only if, it has stored an OCSP Responder key for this RI or LRM.

Transaction Identifier: Allows a Device to provide the RI or LRM with information for tracking of transactions, for example relating to loyalty programs (an example of this could be reward scheme information from the DCF scheme). The Device SHOULD use the <contentID> elements of the ROAP Trigger, when present, to identify the associated DCF(s) from which the TransactionID should be extracted. If no <contentID> elements have been included in the trigger, then the Transaction Identifier SHOULD not be used.

User Domain Authorization: SCE defines this new extension. This extension provides proof to the RI or LRM that the DRM Agent is a member of the User Domain. When sent, this extension MUST be marked critical. The following XML schema fragment defines this extension:

```
<complexType name="UserDomainAuthorizationExtension">
  <complexContent>
    <extension base="roap:Extension">
      <sequence>
        <element name="userDomainAuthorization" type="dom:UserDomainAuthorization"
          maxOccurs =”unbounded”/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Signature is a digital signature on this message.

8.1.2 RO Response

The ROAP-ROResponse message is sent from the RI or LRM to the Device either in response to a ROAP-RORequest message (two-pass variant) or by RI or LRM initiative (one-pass variant). It carries the Protected ROs. In the case of the one-pass variant for a User Domain RO with a <userDomain> constraint but without a *UserDomainContrainedROMoved* extension that would indicate the RO is being Moved to the Device, the RI or LRM MUST have a User Domain Authorization ([SCE-DOM]) for the SCE Device that is valid, not expired, and that indicates the same User Domain generation as that in the RI's or LRM's User Domain Context.

Parameter	ROAP-ROResponse		
	2-pass ***Status = Success***	*2-pass* ***Status ≠ Success***	*1-pass*
status	M	M	M
Device ID	M	-	M
RI ID	M	-	M
Device Nonce	M	-	-
Protected ROs	M	-	M
Certificate Chain	O	-	O
OCSP Response	O	-	M
Extensions	O	-	O
Signature	M	-	M

Table 5 RO Response Message Parameters

status indicates if the request was successfully handled or not. . In the case of the one-pass variant, if a User Domain RO with a <userDomain> constraint is being delivered (but not Moved) and the RI or LRM does not have a current User Domain Authorization for the Device, it can set Status to "UserDomainAuthorizationRequired". Similarly, in the case of the two-pass variant, if a User Domain RO with a <userDomain> constraint is being delivered (but not Moved) and the request does not include a User Domain Authorization for the Device, and the RI or LRM does not have a current one cached, the RI or LRM can set Status to "UserDomainAuthorizationRequired". If an SCE Device receives this error, then it can send an RO Request message that includes its current User Domain Authorization in the User Domain Authorization extension as described above. The Device MAY need to join the User Domain in order to get the UserDomainAuthorization from a DEA (see [SCE-DOM]). In the case of the two-pass variant, if a User Domain RO with a <userDomain> constraint is being delivered (but not Moved) and the request includes an invalid User Domain Authorization, the RI or LRM can set Status to "InvalidUserDomainAuthorization." A User Domain Authorization is invalid if any of these checks fail:

a. verify the DEA signature

b. check whether the <entityId> sub-element matches the *Device ID* value in the request

c. check the <notBefore> and <notAfter> sub-elements to see if the authorization is valid

In the case of the two-pass variant, if a User Domain RO with a <userDomain> constraint is being delivered (but not Moved) and the User Domain generation indicated by the User Domain Authorization for the Device (within the request) is lower than that in the RI's or LRM's User Domain Context, the RI or LRM can set Status to "LowUserDomainGeneration." In the case of the two-pass variant, if the Device's generation is higher than the RI's or LRM's, or the RI or LRM does not have a valid User Domain Authorization for the User Domain, the RI or LRM can set Status to "UserDomainNotSupported" – This applies to User Domain ROs whether or not they have a <userDomain> constraint.

Device ID identifies the requesting Device, in the same manner as in the ROAP-DeviceHello message. The value returned here MUST equal the Device ID sent by the Device in the ROAP-RORequest message that triggered this response in the 2-pass ROAP. In the 1-pass ROAP, the value MUST equal the stored Device ID of the recipient Device. If the Device ID is incorrect, the ROAP-ROResponse processing will fail and the Device MUST discard the received ROResponse PDU.

RI ID identifies the RI or LRM. In the 2-pass protocol, the value MUST equal the RI ID or LRM ID sent by the Device in the preceding ROAP-RORequest message. In the 1-pass protocol, the value MUST equal the stored RI ID or LRM ID.

Device Nonce: This parameter, if present (2-pass), MUST have the same value as the corresponding parameter value in the preceding ROAP-RORequest. If the Device Nonce is incorrect, the ROAP-ROResponse processing will fail and the Device MUST discard the received ROResponse PDU.

Protected RO(s) are the Rights Objects (in the form of <ProtectedRO> elements), in which sensitive information (such as content encryption keys, CEKs) is encrypted. If a Protected RO within an RO Response sent to an SCE DRM Agent contains a User Domain RO with a <userDomain> constraint and the *UserDomainConstrainedROMoved* extension is not present, then the encryption of the REK MUST follow the algorithm used to deliver a Device RO, i.e. the RI or LRM generates the C_1 and C_2 values as per [DRM-DRM-v2.1]. In this case, RI or LRM does not use the algorithm used for a Domain RO. If the *UserDomainConstrainedROMoved* extension is present, then the encryption of the REK MUST follow the algorithm specified in section 13.1.2. For User Domain ROs without a <userDomain> constraint, the REK is encrypted using the algorithm used for a Domain RO using the K_{MAC}, K_{REK} and C value it received from a DEA for the User Domain ([SCE-DOM]). If a Protected RO within the RO Response contains a <moveIndication> element and the *UserDomainConstrainedROMoved* extension is not present, the value of the <originalIssuer> child element MUST equal the RI ID or LRM ID identified by *RI ID*.

Certificate Chain: This parameter MUST be present unless a preceding ROAP-RORequest message contained the Peer Key Identifier extension, the extension was not ignored by the RI or LRM, and its value identified the RI's or LRM's current key. When present, the value of a Certificate Chain parameter shall be as described for the Certificate Chain parameter of the ROAP-RegistrationResponse message

The Device SHOULD check if the RI or LRM certificate chain received in this parameter corresponds to stored certificate verification data for this RI or LRM. If so, the Device does not need to verify the RI or LRM certificate chain again, otherwise the Device MUST verify the RI or LRM certificate chain and MUST compare the hash of the complete DER-encoded subjectPublicKeyInfo component in the received RI or LRM certificate with the RI ID or LRM ID from the request. If an RI or LRM certificate is received that is not in the stored certificate verification data for this RI or LRM, and if the expiry time of the received RI or LRM certificate is later than the RI or LRM Context for this RI or LRM, and the certificate status of the RI or LRM certificate as indicated in the OCSP response is good, then the Device MUST verify the complete chain and SHOULD replace the stored RI or LRM certificate verification data with the received RI or LRM certificate data and set the RI or LRM context expiry time to that of the received RI or LRM certificate expiry time.

OCSP Response: This parameter, when present, SHALL be a complete set of valid OCSP responses for the RI's or LRM's certificate chain. The Device MUST NOT fail due to the presence of more than one OCSP response element. This parameter will not be sent if the Device sent the Extension No OCSP Response in a preceding ROAP-RORequest (and the RI or LRM did not ignore that extension).

Extensions: The following extensions are defined for the ROAP-ROResponse message:

Transaction Identifier: Allows an RI or LRM to provide a Device with information for tracking of transactions, for example relating to loyalty programs (an example of this could be reward scheme information from the DCF). The RI or LRM MUST NOT include a TransactionIdentifier ROAP extension in the ROResponse when the ROResponse contains a RO bound to a GroupID as specified in [DRM-DRM-v2.1] section 10.7, or a parent ID as defined in [DRM-DRM-v2.1] section 10.5. Upon reception of a ROResponse containing a TransactionIdentifier ROAP extension and a RO bound to a GroupID a Device MUST ignore the TransactionIdentifier ROAP extension.

ConfirmROInstallation: Indicates to the DRM Agent that it must confirm installation of the ROs contained in this message by sending a ROAP-ROConfirmRequest PDU to the RI or LRM. RO Confirmation is a critical extension.

UserDomainConstrainedROMoved: SCE defines this new extension. This extension indicates to the Recipient Device that the <userDomain>-constrained RO(s) within the RO Response were Moved from a Source Device to the RI. When sent, this extension MUST be marked critical. The following XML schema fragment defines this extension:

```
<complexType name="UserDomainConstrainedROMoved">
   <complexContent>
      <extension base="roap:Extension"/>
   </complexContent>
</complexType>
```

Signature: This parameter is a digital signature on the data sent in the protocol. The signature is computed using the RI's or LRM's private key and the current message. The signature method is as specified in [DRM-DRM-v2.1] section 5.4.4.2.1. The Device MUST verify the signature.

An RO Response is not valid unless the signature is correct, the RI or LRM certificate chain has been successfully verified, and the OCSP response indicates that the RI or LRM certificate status is good. If the RO Response is not valid, the Device MUST NOT install the received ROs. Before installing any stateful RO (indicated by the stateful attribute of the <ro> element), the Device MUST apply the RO Replay protection described in [DRM-DRM-v2.1] section 10.4. For replay protection of stateless ROs see section 14.1.

8.2 Move Device RO Protocol

The ROAP-MoveDeviceRO protocol enables a DRM Agent to Move its remaining Rights (or part of it) bound to a Device that was originally issued by an RI or LRM to another DRM Agent through the interaction with the RI. In this protocol, the physical distance between two DRM Agents does not matter.

While the OMA DRM 2.1 RO Upload Protocol transfers Rights which had been directly issued from the RI to the requesting DRM Agent, the ROAP-MoveDeviceRO protocol transfers Rights that may have been issued directly from the RI or LRM, or may have been received from other DRM Agents.

8.2.1 MoveDeviceRORequest

The ROAP-MoveDeviceRORequest message is sent from a Source Device to the RI for transferring Rights. The root element of the message MUST be a <moveDeviceRORequest> element of type gen:Request, in which the following elements are present:

element / attribute	usage	value
triggerNonce	O	Default, as specified in [SCE-GEN]
reqID	M	Default, as specified in [SCE-GEN]
resID	M	Default, as specified in [SCE-GEN]
nonce	M	Default, as specified in [SCE-GEN]
time	M	Default, as specified in [SCE-GEN]
certificateChain	O	Default, as specified in [SCE-GEN]
reqInfo	M	Specified below
signature	M	Specified below

Table 6 MoveDeviceRORequest Message Parameters

The <gen:reqInfo> element under the <moveDeviceRORequest> element MUST contain a <sceroap:moveDeviceRORequestInformation> child element as defined by the following XML schema fragment:

```
<element name="moveDeviceRORequest" type="gen:Request" />
  <element name="moveDeviceRORequestInformation">
    <complexType>
      <sequence>
        <element name="rcptDevID" type="gen:Identifier" minOccurs="0" />
        <element name="rightsInfo" type="sceroap:RightsInfo" maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>
```

The <sceroap:moveDeviceRORequestInformation> element in ROAP-MoveDeviceRORequest message includes optional <rcptDevID> element and one or more <rightsInfo> elements.

The <rcptDevID> element represents the identifier of Recipient Device.

The <rightsInfo> element contains information about Rights being transferred to RI.

If there was preceding ROAP-MoveDeviceRO Trigger and its roRequested attribute value was 'true', or the request message is sent without preceding ROAP-MoveDeviceRO Trigger, the <rightsInfo> element within <moveDeviceRORequest> element SHALL include one <rights> element and one <signature> elements. If there was no preceding MoveDeviceRO Trigger or the trigger doesn't include <recipientInfo> element, the <moveDeviceRORequest> element SHALL include one <recipientDeviceID> element. If the trigger includes <recipientInfo> element, since the RI already knows the ID of the Recipient Device, the <moveDeviceRORequest> element SHALL NOT include any <recipientDeviceID> element. For the detail of RightsInfo type, refer to section 7.1.

signature: This element contains a digital signature over the message besides the <signature> element itself. It is made using the negotiated signature algorithm and using the private key of the Source Device.

8.2.2 MoveDeviceROResponse

The ROAP-MoveDeviceROResponse message is sent from the RI to the Source Device as a response against ROAP-MoveDeviceRORequest message. The root element of the message MUST be a <moveDeviceROResponse> element of type gen:Response, in which the following elements are present:

element / attribute	Usage	Value
status	M	Default, as specified in [SCE-GEN]
errorMessage	O	Default, as specified in [SCE-GEN]
errorRedirectURL	O	Default, as specified in [SCE-GEN]
reqID	M	Default, as specified in [SCE-GEN]
resID	M	Default, as specified in [SCE-GEN]
nonce	M	Default, as specified in [SCE-GEN]
certificateChain	O	Default, as specified in [SCE-GEN]
ocspResponse	O	Default, as specified in [SCE-GEN]
resInfo	M	Specified below
signature	M	Specified below

Table 7 MoveDeviceROResponse Message Parameters

The <gen:resInfo> element under the <moveDeviceROResponse> element MUST contain a <sceroap:moveDeviceROResponseInformation> child element as defined by the following XML schema fragment:

```
<element name="moveDeviceROResponse" type="gen:Response" />
  <element name="moveDeviceROResponseInformation">
    <complexType>
      <sequence>
        <element name="prURL" type="anyURI" minOccurs="0" />
      </sequence>
    </complexType>
  </element>
```

The <sceroap:moveDeviceROResponseInformation> element in ROAP-MoveDeviceROResponse message includes optional <prURL> element.

The <prURL> element represents a post response URL that allows an RI to have post-interaction with the user of Source Device using a browsing session. The post-interaction can be user's selection and/or deselection of Rights to be transferred to Recipient Device, billing and charging per amount of transferring Rights and so forth. Further details in the browsing session are beyond the scope of this specification. The value of the <prURL> element MUST be a URL conforming to [RFC2396]. How DRM Agent handles the <prURL> element is as described in section 9.1.3.

signature: This element contains a digital signature over the message besides the <signature> element itself. It is made using the negotiated signature algorithm and using the private key of the RI.

8.3 Move <userDomain>-constrained RO Protocol

The Move <userDomain>-constrained RO Protocol enables a DRM Agent to Move User Domain RO(s) that have the <userDomain> constraint to another DRM Agent via an RI who is associated to the User Domain.

8.3.1 MoveUserDomainConstrainedRORequest

The MoveUserDomainConstrainedRORequest message is sent from a Source Device to an RI for transferring <userDomain>-constrained RO(s) to the RI. The root element of the message MUST be a <MoveUserDomainConstrainedRORequest> element as defined in the following XML schema fragment:

```
<element name="MoveUserDomainConstrainedRORequest" type="gen:Request" />
```

A MoveUserDomainConstrainedRORequest message MUST be formatted as specified in the table below:

element / attribute	usage	Value
triggerNonce	O	Default, as specified in [SCE-GEN]
reqID	M	Source Device's ID
resID	M	RI's ID
nonce	M	Default, as specified in [SCE-GEN]
time	M	Default, as specified in [SCE-GEN]
certificateChain	O	Default, as specified in [SCE-GEN]
reqInfo	M	Specified below
signature	M	Specified below

Table 8 MoveUserDomainConstrainedRORequest Parameters

The <gen:reqInfo> element under the MoveUserDomainConstrainedRORequest message MUST contain a <sceroap**:** moveUserDomainConstrainedRORequestInformation> child element as defined by the following XML schema fragment:

```
<element name="moveUserDomainConstrainedRORequestInformation">
 <complexType>
  <sequence>
   <element name="userDomainConstrainedROInfo" type="sceroap:UserDomainConstrainedROInfo"
maxOccurs="unbounded" />
   <element name="enc_REK" type="xenc:EncryptedKeyType" />
   <element name="sourceDeviceUserDomainAuthorization" type="dom:UserDomainAuthorizationType"
maxOccurs="unbounded" />
   <element name="recipientDeviceId" type="gen:Identifier" minOccurs="0" />
   <element name="mac" type="base64Binary" />
  </sequence>
 </complexType>
</element>
```

```
<complexType name="UserDomainConstrainedROInfo">
 <sequence>
  <element name="rights" type="o-ex:rightsType" />
  <element name="signature" type="ds:SignatureType" />
  <element name="stateInfo" type="o-ex:constraintType" minOccurs="0" maxOccurs="unbounded" />
 </sequence>
</complexType>
```

userDomainConstrainedROInfo: This element carries information about the <userDomain>-constrained RO being transferred to the RI. It contains the following sub-elements:

rights: This element contains the <rights> element of the RO to be Moved.

signature : This element contains the <signature> element of the RO to be Moved.

stateInfo: This element is present if the RO to be Moved is stateful. It carries the Rights to be Moved to the Recipient Device, see section 7.2.

enc_REK: This element contains a wrapped concatenation of a MAC key, K_{MAC}, and one or more protected RO Encryption Key, K_{REK} (more specifically the K_{REK} is XORed with a hash over the <moveIndication> element and the result wrapped with the User Domain Key before concatenation, see section 13.1.2 for details). The protected K_{REK} that is concatenated first is for the first RO (corresponding to the first <userDomainConstrainedROInfo> element in the request); the protected K_{REK} that is concatenated next is for the second RO (corresponding to the second <userDomainConstrainedROInfo> element in the request); etc. The <ds:KeyInfo> element SHALL be the <gen:X509SPKIHash> element, identifying the RI Public Key through the (SHA-1) hash of the DER-encoded subjectPublicKeyInfo value in the RI certificate.

sourceDeviceUserDomainAuthorization: This element provides proof to the RI that the Source Device is a member of the specified User Domain. The value of the <entityId> sub-element must equal that of the <reqID> element. There MUST be one <sourceDeviceUserDomainAuthorization> element per User Domain the RO(s) being Moved are bound to.

recipientDeviceId: This element identifies the Device that is the intended recipient of the RO(s) being Moved. If there is no preceding MoveUserDomainConstrainedROTrigger or the trigger does not include a <recipientInfo> element, this element SHALL be present. If there is a preceding trigger and the trigger includes a <recipientInfo> element, since the RI already knows the ID of the Recipient Device, this element SHALL NOT be present.

mac This element provides integrity protection through a MAC on the canonical version ([SCE-GEN]) of the <reqInfo> element (excluding the <mac> element), using the MAC key, K_{MAC}, wrapped in the <enc_REK> element. The MAC algorithm SHALL be the same algorithm that was negotiated as part of the registration with the RI, i.e. the MAC algorithm stored in the RI Context.

signature: This element contains a digital signature over the message besides the <signature> element itself. It is made using the negotiated signature algorithm and using the private key of the Source Device.

Upon receipt of a MoveUserDomainConstrainedRORequest message, the RI MUST perform the processing specified in section 9.2.2.

8.3.2 MoveUserDomainConstrainedROResponse

The MoveUserDomainConstrainedROResponse message is sent from the RI to the Source Device as a response to the MoveUserDomainConstrainedRORequest message. The root element of the message MUST be a <MoveUserDomainConstrainedROResponse> element as defined in the following XML schema fragment:

```
<element name="moveUserDomainConstrainedROResponse" type="gen:Response" />
```

If the processing of the MoveUserDomainConstrainedRORequest is not successful, then the response MUST be formatted as specified in [SCE-GEN]. Otherwise the response MUST be formatted per the table below:

element / attribute	usage	value

status	M	”Success”
errorMessage	O	Default, as specified in [SCE-GEN]
reqID	M	Source Device’s ID
resID	M	RI’s ID
nonce	M	Default, as specified in [SCE-GEN]
certificateChain	O	Default, as specified in [SCE-GEN]
ocspResponse	O	Default, as specified in [SCE-GEN]
resInfo	M	Specified below
signature	M	Specified below

Table 9 MoveUserDomainConstrainedROResponse Message Parameters

The <gen:resInfo> element under the MoveUserDomainConstrainedROResponse message MUST contain a <sceroap:moveUserDomainConstrainedROResponseInformation> child element as defined by the following XML schema fragment:

```
<element name="moveUserDomainConstrainedROResponseInformation">
 <complexType>
  <sequence>
   <element name="RIUserDomainAuthorization" type="dom:UserDomainAuthorizationType"
maxOccurs="unbounded" />
  </sequence>
 </complexType>
</element>
```

RIUserDomainAuthorization: This element provides proof to the Source Device that the RI is authorized to manage the specified User Domain. There MUST be one <RIUserDomainAuthorization> element per User Domain the RO(s) being Moved are bound to.

signature: This element contains a digital signature over the message besides the <signature> element itself. It is made using the negotiated signature algorithm and using the private key of the RI.

8.4 RO Upgrade

8.4.1 RO Upgrade Request

The ROAP-ROUpgrade Request message is sent from the Device to the RI to request upgrading one or more existing ROs in the Device. The root element of the message MUST be a <roUpgradeRequest> element of type gen:Request, in which the following elements are present:

element / attribute	usage	Value
triggerNonce	O	Default, as specified in [SCE-GEN]
reqID	M	Default, as specified in [SCE-GEN]
resID	M	Default, as specified in [SCE-GEN]
nonce	M	Default, as specified in [SCE-GEN]
time	M	Default, as specified in [SCE-GEN]
certificateChain	O	Default, as specified in [SCE-GEN]
reqInfo	M	Specified below
signature	M	Specified below

Table 10 ROUpgradeRequest Message Parameters

The <gen:reqInfo> element under the <roUpgradeRequest> element MUST contain a <sceroap:roUpgradeRequestInformation> child element as defined by the following XML schema fragment:

```
<element name="roUpgradeRequest" type="gen:Request" />
 <element name="roUpgradeRequestInformation">
  <complexType>
   <sequence maxOccurs="unbounded">
    <element name="existingRights" type="sceroap:RightsInfo" />
    <element name="ROUpgradeInfo" type="base64Binary" minOccurs="0" />
   </sequence>
  </complexType>
 </element>
```

The <sceroap:roUpgradeRequestInformation> element in RO Upgrade Request message includes one or more sequence of one <existingRights> element and optional <upgradeInfo> element.

The <existingRights> element includes the information specifying the existing ROs, corresponding state information (if stateful) and REK etc.

The <ROUpgradeInfo> element is present if the preceding trigger does not contain <upgradeInfo> element. It contains the user's wish of additional permission and/or modified constraint(s) of existing permission(s) for the existing ROs indicated by <existingRights> element.

If there was preceding ROAP-ROUpgrade Trigger and its roRequested attribute value was 'true', or the request message is sent without preceding ROAP-ROUpgrade Trigger, The <existingRights> element of type sceroap:RightsInfo SHALL include <rights> and <signature> elements. For the detail of RightsInfo type, refer to section 7.1.

signature: This element contains a digital signature over the message besides the <signature> element itself. It is made using the negotiated signature algorithm and using the private key of the Source Device.

8.4.2 RO Upgrade Response

The ROAP-ROUpgradeResponse message is sent from the RI to the Device in response to a ROAP-ROUpgradeRequest message. The root element of the message MUST be a <roUpgradeResponse> element of type gen:Response, in which the following elements are present:

element / attribute	usage	value
status	M	Default, as specified in [SCE-GEN]
errorMessage	O	Default, as specified in [SCE-GEN]
errorRedirectURL	O	Default, as specified in [SCE-GEN]
reqID	M	Default, as specified in [SCE-GEN]
resID	M	Default, as specified in [SCE-GEN]
nonce	M	Default, as specified in [SCE-GEN]
certificateChain	O	Default, as specified in [SCE-GEN]
ocspResponse	O	Default, as specified in [SCE-GEN]
resInfo	M	Specified below
signature	M	Specified below

Table 11 ROUpgradeResponse Message Parameters

The <gen:resInfo> element under the <roUpgradeResponse> element MUST contain a <sceroap:roUpgradeResponseInformation> child element as defined by the following XML schema fragment:

```
<element name="roUpgradeResponse" type="gen:Response" />
 <element name="roUpgradeResponseInformation">
```

```
  <complexType>
   <sequence>
    <element name="upgradeResult" type="sceroap:UpgradeResult" maxOccurs="unbounded" />
   </sequence>
  </complexType>
</element>

<complexType name="UpgradeResult">
  <sequence>
   <element name="existingROID" type="ID" />
   <choice>
    <element name="newRO" type="roap:ProtectedRO" />
    <element name="failureReason" type="string" />
   </choice>
  </sequence>
</complexType>
```

The <sceroap:roUpgradeResponseInformation> element in RO Upgrade Response message includes one or more <upgradeResult> element.

The <upgradeResult> element contains one <existingROID> element and either a <newRO> element (of roap:ProtectedRO type) or a <failureReason> element. If the RO corresponding to the <existingROID> element is successfully upgraded, then the <newRO> element is included, else the <failureReason> element is included.

The <newRO> element contains the RO with the upgraded permissions and/or constraints from the RI.

signature: This element contains a digital signature over the message besides the <signature> element itself. It is made using the negotiated signature algorithm and using the private key of the RI.

9. Moving RO via an RI

The protocol implementation is required only for the Source Device, and the Recipient Device simply uses existing RO Acquisition Protocol as per section 8.1. How to Move DRM 2.x Domain RO and Parent RO is beyond the scope of this version of specification.

9.1 Moving Device RO

9.1.1 Sending MoveDeviceRORequest

ROAP-MoveDeviceRO protocol can be initiated either by receiving a ROAP-MoveDeviceRO Trigger or by user interaction with the Device (e.g. the user of the Source Device can select RI-issued Rights to Move using a built-in menu in the phone).

To package a ROAP-MoveDeviceRORequest message, the DRM Agent MUST proceed as follows:

1. The Device lets the user select Device Rights Objects that are issued by an RI or LRM to be Moved. The Source Device MUST ensure that the selected Rights Object has a <move> permission containing no <system> constraint or a <move> permission containing a <system> constraint which identifies Move Device RO via RI protocol. Further details of this step are beyond the scope of this specification.

2. The DRM Agent marks the selected Rights Objects as unusable. If the Rights Object is stateful and just a portion of the Rights Object is being Moved, then it marks the portion being Moved as unusable.

3. The DRM Agent generates a ROAP-MoveDeviceRORequest message which includes one or more <rightsInfo> elements. Generation of <rightsInfo> element (of sceroap:RightsInfo type) conforms to section 7.1. If the RO included in the <rightsInfo> element is created by an RI, the DRM Agent MUST ensure that the RI (as peer entity of recipient of MoveDeviceRORequest message) is the same RI which is indicated in the <signature> element in the <rightInfo> element. If the RO included in the <rightsInfo> element is created by an LRM, the DRM Agent MUST ensure that the RI (as peer entity of recipient of MoveDeviceRORequest message) is one of the RIs that is indicated within the RO as being eligible to provide Move service.

4. If there was a preceding trigger, the DRM Agent sends the request message using the roapURL in the trigger message. Else, the DRM Agent sends the request message to the riURL which is stored in the RI Context.

If any error occurred during sending the request message, the DRM Agent MAY resend the request message. When resending the request message, the DRM Agent SHALL use same nonce with previous request message and use current DRM Time as <time> element. How many times the DRM Agent retries is left to implementation.

9.1.2 Processing MoveDeviceRORequest

When an RI receives a MoveDeviceRORequest message, the RI MUST process the request message as follows:

1. it checks if it has valid Device Context with the Device sending the request message by checking the value of <reqID> element of the ROAP-MoveDeviceRORequest message. If the Device Context is unavailable or invalid e.g. expired, the RI MUST respond with *NotRegistered* error and abort the process.

2. it verifies the <signature> element in the request message. The signature verification conforms to [DRM-DRM-v2.1]. If the verification is not successful, the RI MUST respond with appropriate error (i.e. *SignatureError, NoCertificateChain, InvalidCertificateChain or TrustedRootCertificateNotPresent*) and abort the process.

3. it checks the <nonce> element in the request message according to section 15.

4. it checks the value of <time> element in the request message. Processing of the value of <time> element conforms to [DRM-DRM-v2.1]. If the DRM Agent has invalid DRM Time, the RI MUST respond with *RequesterTimeError* error and abort the process.

5. it verifies each <rightsInfo> element in the request message (see section 7.1.2). Additionally it MUST check that the <rights> element in the <rightsInfo> element has a <move> permission that does not preclude Moving Device ROs via the RI (i.e. having no <system> constraint on the <move> permission or having a <system> constraint which identifies Move Device RO via RI protocol on the <move> permission). If it does not, the RI MUST respond with *MovePermissionNotPresent* error and abort the process.

6. if all above steps were successful, it responds with a MoveDeviceROResponse that contains the <status> element that has "Success" value.

7. it generates ROs cryptographically bound to the Recipient Device, based on the received <rights> element and their corresponding State Information.

When the RI generates the ROs for the recipient Device, the RI SHALL set the value of stateful constraint in the <rights> element to the value given by the corresponding <stateInfo> element in the request message. If the <rights> element has "count" constraint under "move" permission, the RI SHALL decrease the value of the <o-dd:count> element under "move" permission by 1. The RI SHALL use new RO Encryption Key to encrypt Content Encryption Key constructing the <KeyInfo> element (under <asset> element). The RI SHALL NOT put into the new RO the <moveIndication> element if one was received in the request. (Note: if the original issuer of the RO is an LRM and RO has a LRM signature, then the RO in the request message SHALL contain the <moveIndication> element. If the original issuer of the RO is the RI, then the RO in the request message SHALL NOT contain the <moveIndication> element.) After that, the RI MUST add a <signature> element which contains signature value over the <rights> element.

8. it conducts a typical 1-pass or 2-pass RO acquisition protocol or 4-pass confirmed RO acquisition protocol as per section 8.1to issue generated ROs to the Recipient Device. In case of 2-pass RO acquisition protocol or 4-pass confirmed RO acquisition protocol, the RI sends an ROAP trigger to the recipient device in order to instruct the recipient device to download the Rights Object generated by RI which is based on the one previously transferred from the source device.

How the RI handles for the case that RI fails to issue the Rights Objects to the Recipient Device is beyond the scope of this specification.

9.1.3 Processing MoveDeviceROResponse

When a DRM Agent receives a ROAP-MoveDeviceROResponse message, the DRM Agent MUST process the response message as follows:

1. it checks if <reqID>, <resID>, <nonce> elements in the response message are same as the preceding request message. If any of these does not match, it terminates the MoveDevice RO protocol.

2. it verifies <signature> element in the response message. If the verification is failed, it terminates the ROAP-MoveDeviceRO protocol.

3. it checks <status> element in the response message and process as follows:

   a. If the status in the response message is "Success", the DRM Agent MUST:

   - In case of Move of full Rights, remove the corresponding ROs and (and their State Information if present) which were identified in the request message.

   - In case of Move of partial Rights, update their State Information by amount of transferred rights. E.g. if 3 counts remained before starting the ROAP-MoveDeviceRO protocol and 1 count was transferred to RI, then the DRM Agent decrements the State Information to be 2 counts.

- If the Post Response URL extension is present, the DRM Agent MUST send an HTTP GET request to the URL specified in the value of the <prURL> element of this extension at the first available opportunity. If the request fails with error code 404 Not Found [RFC2616], the Device SHOULD NOT make further requests to the URL. If the request fails with some other error, the Device MAY retry the request at a later time.

b. If the status in the response message is not "Success", the DRM Agent MUST mark the corresponding ROs as usable and terminate the ROAP-MoveDeviceRO protocol.

9.2 Moving <userDomain>-constrained RO

9.2.1 Sending MoveUserDomainConstrainedRORequest

The Move <userDomain>-constrained RO protocol can be initiated either by receiving a MoveUserDomainConstrainedROTrigger or by user interaction with the Device (e.g. the User of the Source Device can select RO(s) to Move using a built-in menu in the phone).

To package a MoveUserDomainConstrainedRORequest message, the DRM Agent MUST proceed as follows:

1. let the User select <userDomain>-constrained RO(s) to be Moved. The DRM Agent MUST ensure that the selected RO(s) have a <move> permission containing no <system> constraint or a <move> permission containing a <system> constraint which identifies the Move <userDomain>-constrained RO via RI protocol. The DRM Agent MUST also ensure the RI´s ID is present in the <moveIndication> element.
2. mark the selected RO(s) as unusable. For stateful RO, if only a portion of the Rights is to be Moved, only that portion is marked as unusable.
3. generate a MoveUserDomainConstrainedRORequest message. For each of the User Domains the selected RO(s) are bound to, a corresponding <sourceDeviceUserDomainAuthorization> element MUST be included.
4. if there was a preceding MoveUserDomainConstrainedRO trigger, the DRM Agent sends the request message using the reqURL in the trigger. Else, the DRM Agent sends the request message to the riURL which is stored in the RI Context.

If any error occurred during sending the request message, the DRM Agent MAY resend the request message. When resending the request message, the DRM Agent SHALL use same nonce with previous request message and use current DRM Time as <time> element. How many times the DRM Agent retries is left to implementation.

9.2.2 Processing MoveUserDomainConstrainedRORequest

When an RI receives a MoveUserDomainConstrainedRORequest message, the RI MUST process the request message as follows:

1. check if there is a valid Device Context by checking the value of <reqID> element of the request message. If the Device Context is unavailable or invalid e.g. expired, the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "NotRegistered" and abort the process.
2. verify the <signature> element in the request message. If the verification is not successful, the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "SignatureError" and abort the process.
3. it checks the <nonce> element in the request message according to section 15.

4. it checks the value of <time> element in the request message. Processing of the value of <time> element conforms to [DRM-DRM-v2.1]. If the DRM Agent has invalid DRM Time, the RI MUST respond with *RequesterTimeError* error and abort the process.

5. verify the <mac> element as follows:

   a. unwrap <enc_REK> to recover K_{MAC} (see section 13.1.2).

   b. calculate a MAC on the canonical version of the <reqInfo> element (excluding the <mac> element) using the K_{MAC}. The MAC algorithm to use is defined in the Device Context.

   c. check the calculated MAC value against the <mac> element. If the calculated value is not equal to the value of the <mac> element, the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "MACError" and abort the process.

6. for each <userDomainConstrainedROInfo> element in the request message:

   a. the RI SHOULD check for the presence of its own ID in the <moveIndication> element. If not present, the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "MoveServiceNotProvided" and abort the process.

   b. verify the <signature> sub-element and check whether the signature has been generated by the RI/LRM that is identified by the <userDomainAuthorization> element under <rights>

   c. validate the <userDomainAuthorization> element under <rights>

   d. check if the <rights> sub-element has the <move> permission that does not preclude the use of the Move <userDomain>-constrained RO via RI protocol (i.e. having no <system> constraint on the <move> permission or having a <system> constraint which identifies the Move <userDomain>-constrained RO via RI protocol)

   e. if the <stateInfo> sub-element is present, validate that the state information contained in <stateInfo> is consistent with the original state in the <rights> element

   If any of the b-e checks fails, the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "InvalidRO" and abort the process. Note that the RI May provide more detailed error information in the response by using the <errorMessage> attribute.

   If for some reason (e.g. the RI doesn't trust some RI/LRM(s) identified by the <moveIndication> element, the RI decides to not provide the Move service, the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "MoveServiceNotProvided" and abort the process.

7. for each <sourceDeviceUserDomainAuthorization> element in the request message:

   a. verify the DEA signature

   b. check whether the <entityId> sub-element matches the <reqID> element in the request

   c. check the <notBefore> and <notAfter> sub-elements to see if the authorization is valid

   If any of the above checks fails, the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "InvalidUserDomainAuthorization" and abort the process. Otherwise, the RI checks whether itself has a valid User Domain Authorization issued by the DEA for this User Domain. If not the RI MUST send a response with the *status* attribute set to "UserDomainNotSupported" and abort the process.

   d. check whether the generation number in the <userDomainId> element is the same as the one in RI's User Domain Context for that particular User Domain. If the Source Device's generation is lower than RI's, the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "LowUserDomainGeneration" and abort the process. If the Source Device's generation is higher than RI's,

the RI MUST send to the Source Device a MoveUserDomainConstrainedROResponse message with the *status* attribute set to "UserDomainNotSupported" and abort the process.

e. If in the request message there is no <sourceDeviceUserDomainAuthorization> element for a User Domain to which one or more of the RO(s) being Moved are bound, the RI MUST send a response with the *status* attribute set to "UserDomainAuthorizationRequired" and abort the process.

If all the above steps are successful, the RI SHALL set the *status* attribute in the MoveUserDomainConstrainedROResponse message to "Success". In this case, the RI SHALL also include a list of <RIUserDomainAuthorization> elements in the response message, one per User Domain the ROs being Moved are bound to.

When and only when all the <userDomainConstrainedROInfo> elements in the request have been successfully validated (including checking of the required User Domain Authorizations), the RI generates corresponding ROs for the Recipient Device. The RI SHALL use the key received in the <enc_REK> element as the RO Encryption Key (see section 13.1.2). If the RO is stateful, the RI SHALL set the value of stateful constraint in the <rights> element to the value given by the corresponding <stateInfo> element in the request message. If the <rights> element has a "count" constraint under the "move" permission, the RI SHALL decrease the value of the <o-dd:count> element by 1. The RI SHALL also put a <userDomainAuthorization> element into the <rights> element proving that it is authorized by the DEA for managing the User Domain. After that, the RI MUST generate a signature over the <rights> element. The RI SHALL put into the new RO the <moveIndication> element that was received in the request.

The RI conducts a 1-pass or 2-pass RO Acquisition protocol or a 4-pass Confirmed RO Acquisition protocol ([DRM-DRM-v2.1]) to issue generated RO(s) to the Recipient Device. The RI MUST include a *UserDomainConstrainedROMoved* extension in the RO Response.

How the RI handles the case that it failed to issue the Rights Objects to the Recipient Device is beyond the scope of this specification.

9.2.3 Processing MoveUserDomainConstrainedROResponse

When a DRM Agent receives a MoveUserDomainConstrainedROResponse message, the DRM Agent MUST process the response message as follows:

1. check if the <reqID>, <resID>, <nonce> elements in the response message match the ones sent in the preceding request message. If any of these does not match, the DRM Agent SHALL terminate the Move <userDomain>-constrained RO protocol.

2. verify the <signature> element in the response message. If the verification fails, the DRM Agent SHALL terminate the Move <userDomain>-constrained RO protocol.

3. check the <status> element in the response message. If the value is not "Success", the DRM Agent SHALL mark all the RO(s) being Moved as usable and terminate the Move <userDomain>-constrained RO protocol.

4. for each <RIUserDomainAuthorization> element in the response message:

   a. verify the DEA signature

   b. check whether the <entityId> sub-element matches the <resID> element in the response

   c. check the <notBefore> and <notAfter> sub-elements to see if the authorization is valid

   If any of the above checks fails, the DRM Agent SHALL mark those RO(s) that are bound to this User Domain as usable.

   If in the response message there is no <RIUserDomainAuthorization> element for a User Domain to which one or more of the RO(s) being Moved are bound, the DRM Agent SHALL mark those ROs as usable.

5. for each of the rest of RO(s) being Moved:

- In case of Move of full Rights, remove the RO and associated State Information
- In case of Move of partial Rights, update the associated State Information by the amount of transferred rights. E.g. if 3 counts were remained before starting the Move <userDomain>-constrained RO protocol and 1 count was transferred to RI, then the DRM Agent updates the State Information to have 2 counts.

10.Upgrading Rights

ROAP-ROUpgrade protocol enables a Device to obtain from RI additional permissions for an existing RO (which does not have these permissions) that was issued from the same RI. For example, if a User owns an RO which does not include <move> permission and hence is not allowed to be Moved directly to another Device, then by RO Upgrade protocol, the User can obtain from the RI a <move> permission for this RO. Under the control of the <move> permission, the RO and its State Information can be Moved directly between Devices.

How the message in this protocol is sent and processed is described in the sections below.

10.1 Sending ROUpgradeRequest

The ROUpgrade protocol MAY be initiated by an *RO Upgrade* trigger which carries the <roUpgradeTrgInformation> element that may carry the <roID> element , the <upgradeInfo> element and "roRequested" attribute. When a Device receives an *RO Upgrade trigger*, it constructs an ROUpgradeRequest message as per the trigger:

- If <roUpgradeTrgInformation> element carries one or more <roID> elements, the RO(s) to be Upgraded are specified by the RI and hence the Device SHALL NOT request an Upgrade for any other RO(s) than the RO(s) specified by <roID> element(s). Else, it means that the RO(s) to be Upgraded are not specified by the RI and the Device MAY request to Upgrade any valid RO the user wants.
- If the <roUpgradeTrgInformation> element carries an <upgradeInfo> element, the Device SHALL put the nonce of RO Upgrade Trigger into the TriggerNonce attribute of the <roUpgradeRequest> element in the ROAP-ROUpgradeRequest message. Else, the Device SHALL put user's wish of additional permission(s) and/or modified constraint(s) of existing permission(s) into the <ROUpgradeInfo> element in the ROUpgradeRequest message.
- If the <roUpgradeTrgInformation> element carries an "roRequested" attribute set to "false" the Device SHALL not put the <rights> and <signature> elements into <existingRights> (of sceroap:RightsInfo type) in ROAP-ROUpgradeRequest message. If the value of "roRequested" attribute is not present or is set to "true", the Device receiving *RO Upgrade* trigger SHALL put the <rights> element and <signature> element related to the existing RO(s) that it attempts to Upgrade into the <existingRights> element (of sceroap:RightsInfo type) in ROAP-ROUpgradeRequest message. See section 7.1.1 for more details on how to package a <existingRights> element of sceroap:RightsInfo type. Else the Device SHALL not put <rights> element and <signature> element into <existingRights> element in ROAP-ROUpgradeRequest message.

The ROUpgrade protocol MAY also be initiated without an *RO Upgrade* trigger: for example, the user can operate on some user interface to initiate an upgrade for an existing RO. In this case, the Device MUST include the original RO that it is attempting to Upgrade into the ROUpgrade Request message. During the construction of the ROUpgradeRequest message, the Device MAY form <ROUpgradeInfo> element according to the user's operation on some user interface. What the user interface is and how the user operates on it to designate what additional permission he/she needs is out of scope of OMA DRM.

After generating the ROAP-ROUpgradeRequest according to the message syntax and the rules mentioned above, the DRM Agent MUST marks the RO that it attempts to Upgrade as unusable. After that, the DRM Agent sends the generated ROAP-ROUpgradeRequest message to an URL indicated by the <reqURL> element in preceding trigger. If there was no preceding trigger, the DRM Agent sends the generated request message to the riURL, which is stored in the RI Context.

If transport-level error occurred during sending the request message or receiving the response message, the Device MAY resend the message. How many times the DRM Agent retries is left to implementation. In case of final failure, the DRM Agent MUST mark the Rights Object as usable.

10.2 Processing ROUpgradeRequest

When an RI receives an ROUpgradeRequest message, the RI MUST process the request message as follows:

1. it checks if it has valid Device Context with the Device that sends the ROUpgradeRequest message. If RI finds that the Device Context corresponding to the <deviceID> element of the ROAP-ROUpgradeRequest message is unavailable or invalid e.g. expired, it MUST respond with *NotRegistered* error. RI MAY form an <errorMessage> element that includes the textual reason for this failure and put it into the response message.

2. it verifies the Device's signature on the whole message, using the last <signature> element in the message. The signature verification conforms to [DRM-DRM-v2.1]. If the verification is not successful, the RI MUST respond with *SignatureError* error. RI MAY form an <errorMessage> element that includes the textual reason for this failure and put it into the response message

3. it checks the value of <time> element in the request message. Processing of the value of <time> element conforms to [DRM-DRM-v2.1]. If the RI detects that the DRM Agent has invalid DRM Time, the RI MUST respond with *RequesterTimeError* error. RI MAY form an <errorMessage> element that includes the textual reason for this failure and put it into the response message

4. it verifies the RI's signature on the <rights> element in the request message, using the <signature> element under the <rightsInfo> element. The verification must include the step for checking whether the signature was generated by the RI. If the verification fails, the RI MUST respond with *UnknownRO* error. RI MAY form an <errorMessage> element that includes the textual reason for this failure and put it into the response message

5. if any <stateInfo> element is supplied under the <rightsInfo> element in the request message, check whether the State Information, indicates that the Rights is still available for the Device that sends the request message, in case the additional permission is about sharing (e.g. Move or Copy) permission. If RI finds the Rights is already not available for the Device, then the RI MUST respond with *InvalidRO* error. RI MAY form an <errorMessage> element that includes the textual reason for this failure and put it into the response message

6. The RI responds with an ROUpgradeResponse that contains:

   - <status> element with "Success" value
   - new upgraded RO as the <newRO> element. The <rights> in the new RO is created based on the <existingRights> element which is from an existing RO and the user's wish of additional permission(s) and/or modified constraint(s) of existing permission(s) indicated by the <ROUpgradeInfo> element in the corresponding ROUpgradeRequest message or by a previous browsing session.

If for some reason the RI decides to not provide the Upgrade service, the RI MUST send to the Source Device a ROUpgradeResponse message with the status attribute set to "UpgradeServiceNotProvided".

10.3 Processing ROUpgradeResponse

When the DRM Agent receives an ROUpgradeResponse message, it MUST process the response message as follows:

1. it verifies the RI's signature on the response message. The signature verification conforms to [DRM-DRM-v2.1]. If the signature verification was not successful, it MUST mark the corresponding RO as usable and terminate RO Upgrade protocol.

2. it checks <status> element in the response message as follows:

   - If the status equals "Success",
     - the DRM Agent checks the <existingROID> element in response message. If the RO indicated by the <existingROID> is not present or not marked as unusable, then the DRM Agent terminates RO Upgrade protocol.
     - the DRM Agent removes permanently the corresponding RO.
     - the DRM Agent installs the new upgraded RO.indicated by the <newRO> element in the response message.

- If the status does not equal “Success”, then the DRM Agent MUST mark the corresponding RO as usable. The Device MAY notify the reason for failure to the User of the DRM Agent using the <errorMessage> in the response message.

11.Installation of an RO

11.1 RO with new SCE permissions

Many SCE protocols require the XML RO to be transferred. DRM Agents should be aware that upon installation of an RO, it may need to store the XML RO or otherwise be able to recover the XML RO to be used in SCE protocols.

11.2 RO with <party> element

When installing an RO which was not received via a Move RO transaction or Copy RO operation or Share RO operation or Lend RO operation (all of which are described in [SCE-A2A]), the DRM Agent MUST follow the procedures described in [DRM-DRM-v2.1] section 10.3 and MUST perform the following checks before installation:

1. If the <agreement> element contains a <party> element, in which the value of an <uid> element in the <context> element has the form "device:*x*" (without the quotes), the DRM Agent MUST verify that the value of *x* equals the base64 encoded SHA-1 hash over the concatenation of the ROID and the DRM Agent's Device ID.
2. If the <agreement> element contains a <party> element, in which the value in an <uid> element in the <context> element has the form "dom:*x*:*y*" (without the quotes):
    a. The DRM Agent MUST verify that the value of *x* equals the RI ID in the RO.
    b. The DRM Agent MUST verify that the value of *y* equals the SHA-1 hash over the concatenation of the ROID and the Domain ID of the Domain to which the RO is bound.
3. If the <agreement> element contains a <party> element, in which the value in an <uid> element in the <context> element has the form "udom:*x*:*y*" (without the quotes):
    a. The DRM Agent MUST verify that the value of *x* equals the DEA ID in the RO.
    b. The DRM Agent MUST verify that the value of *y* equals the SHA-1 hash over the concatenation of the ROID and the User Domain ID of the User Domain to which the RO is bound.
4. If the <agreement> element contains a <party> element with a <date> element, the DRM Agent MUST verify that the value of the <fixed> element in this <date> element equals the value of the <timeStamp> element in the <ro> element.

If check 2 or check 3 failed, the DRM Agent MAY try to join the Domain and perform the check again.

The DRM Agent MUST NOT install the RO if any of the four checks failed.

12.Transport Mapping

12.1 HTTP Transport Mapping

12.1.1 Multiple ROs Acquisition Triggered by DCF Headers

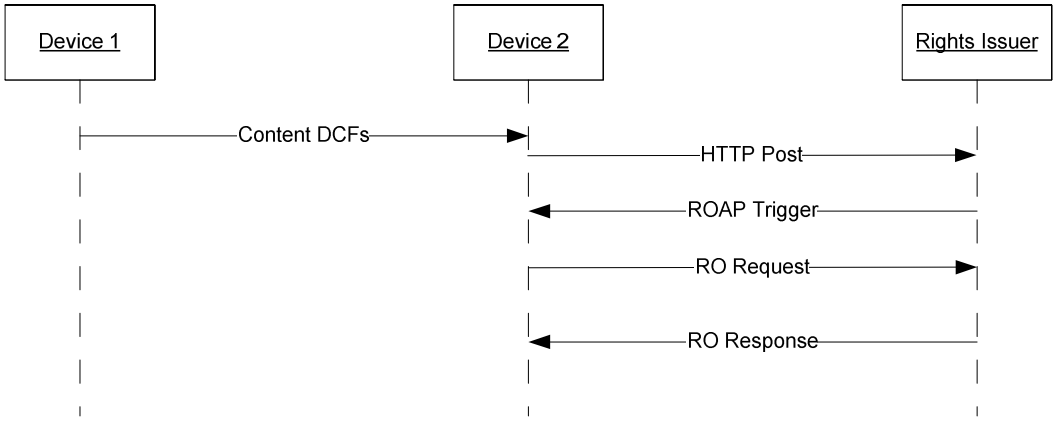

Figure 4 Multiple ROs Acquisition Triggered by DCF Headers

In this case multiple DCFs are superdistributed to a Device, and the DRM Agent uses DCF headers to initiate a ROAP transaction and download multiple Rights Objects.

– A user receives multiple DCFs from another Device, e.g. through MMS, peer-to-peer, removable media, or some other transfer mechanism.

– If multiple DCFs contain the same BatchRIURL header, then the DRM Agent attempts to send a ROAP Trigger request to the RI or LRM. If the DRM Agent has an existing RI or LRM Context for the RI or LRM, then the DRM Agent may send an HTTP Post to the BatchRIURL.

```
The DRM Agent sends an HTTP Post to the URL specified by the BatchRIURL
header.  POST /ro.cgi
Host: www.acme.com

cid1=qw683hgew7d
cid2=qw724eldw5d
```

– The RI or LRM returns an HTTP response containing a ROAP Trigger.

```
HTTP 1.1 200 OK
 Server: CoolServer/1.3.12
 Content-Length: 986
 Content-Type: application/vnd.oma.drm.roap-trigger+xml

... [ROAP Trigger] ...
```

– The ROAP Trigger is used by the DRM Agent on the Device to initiate a ROAP session to download multiple Rights Objects. The POST includes a ROAP-RORequest PDU in the HTTP request body.

```
POST http://www.acme.com/ro.cgi
```

```
 Host: www.acme.com
 User-Agent: CoolPhone/1.4
 Accept: application/vnd.oma.drm.roap-pdu+xml, multipart/related
 Accept-Charset: utf-8
 Content-Length: 125
 Content-Type: application/vnd.oma.drm.roap-pdu+xml

... [ROAP PDU] ...
```

A previously established RI or LRM Context is assumed in the example. If this were not the case, then the ROAP-RORequest would be preceded by a ROAP-Registration transaction.

– The RI or LRM returns an HTTP response containing a ROAP-ROResponse PDU in the HTTP response body.

```
HTTP 1.1 200 OK
 Server: CoolServer/1.3.12
 Content-Length: 986
 Content-Type: application/vnd.oma.drm.roap-pdu+xml

... [ROAP PDU] ...
```

13.Key Management

13.1 Key Transport Mechanisms

13.1.1 Distributing KMAC and KREK under an RI Public Key

This section applies when encrypting RO Encryption Key and MAC Key for the Move Device RO via RI protocol or the RO Upgrade protocol.

K_{REK} ("Rights Object Encryption Key") is the 128-bit wrapping key for the content-encryption key K_{CEK} in Rights Objects. In the Move Device RO and RO Upgrade protocols, the K_{REK} transferred in the <encKey> element of <rightsInfo> MUST be the same as the one from the original Rights Object. K_{MAC} is a 128-bit long key generated randomly by the sender and used for key confirmation of the message carrying K_{REK}.

The asymmetric encryption scheme RSAES-KEM-KWS shall be used with the AES-WRAP symmetric-key wrapping scheme to securely transmit K_{MAC} and K_{REK} to a recipient RI using the RI's RSA public key. An independent random value Z shall be chosen for each encryption operation. For the AES-WRAP scheme, K_{MAC} and K_{REK} are concatenated to form K, i.e.:

KEK = KDF(I2OSP(Z, mLen), **NULL**, kekLen)

C_2 = AES-WRAP(KEK, K_{MAC} | K_{REK})

C_1 = I2OSP(RSA.ENCRYPT($PubKey_{RI}$, Z), mLen)

$C = C_1 | C_2$

where kekLen shall be set to 16 (128 bits) and mLen is the length of the modulus of the RI's RSA public key in octets. In this way, AES-WRAP is used to wrap 256 bits of key data (K_{MAC} | K_{REK}) with a 128-bit key-encryption key (KEK).

After receiving C, the RI splits it into C_1 and C_2 and decrypts C_1 using its private key (consisting of a private exponent d and the modulus m), yielding Z:

$C_1 | C_2 = C$

c_1 = OS2IP(C_1, mLen)

Z = RSA.DECRYPT($PrivKey_{RI}$, c_1) = c_1^d mod m

where OS2IP converts an octet string to a nonnegative integer and is defined in [PKCS-1].

Using Z, the RI can derive KEK, and from KEK unwrap C_2 to yield K_{MAC} and K_{REK}.:

KEK = KDF(I2OSP(Z, mLen), **NULL**, kekLen)

K_{MAC} | K_{REK} = AES-UNWRAP(KEK, C_2)

The following URI shall be used to identify this key transport scheme in <xenc:EncryptionMethod> elements:

http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-1#rsaes-kem-kdf2-kw-aes128

13.1.2 Transporting KMAC and one or more Protected KREK under a RI Public Key

This section applies to the Move <userDomain>-constrained RO via RI protocol.

K_{REK} ("Rights Object Encryption Key") is the wrapping key for the content-encryption key K_{CEK} in Rights Objects. To prevent the RI from getting knowledge of K_{REK} when Moving a <userDomain>-constrained RO via the RI, K_{REK} is protected by using the User Domain Key (UDK) before transferred to the RI.

K_{MAC} is a 128-bit long key generated randomly by the Source Device and used for key confirmation of the message carrying one or more protected K_{REK}.

Let Hash16 denote the first 16 bytes of the SHA-1 hash of the <moveIndication> element. K_{REK} is first XORed with Hash16 and the result then wrapped with the UDK using the AES-WRAP symmetric wrapping scheme [AES-WRAP]. Let PREK denote the protected K_{REK}:

PREK = AES-WRAP(UDK, K_{REK} XOR Hash16)

The asymmetric encryption scheme RSAES-KEM-KWS SHALL be used with the AES-WRAP symmetric-key wrapping scheme to securely transmit K_{MAC} and one or more PREK to a recipient RI using the RI's RSA public key. An independent random value Z SHALL be chosen for each encryption operation. For the AES-WRAP scheme, K_{MAC} and one or more PREK are concatenated to form K, i.e.:

KEK = KDF(I2OSP(Z, mLen), **NULL**, kekLen)

C_2 = AES-WRAP(KEK, K_{MAC} | $PREK_1$ | … | $PREK_n$) (*n* is the number of ROs being Moved, the PREKs are concatenated in the same order as the corresponding <userDomainConstrainedROInfo> elements appear in the request)

C_1 = I2OSP(RSA.ENCRYPT($PubKey_{RI}$, Z), mLen)

C = C_1 | C_2

where kekLen SHALL be set to 16 (128 bits) and mLen is the length of the modulus of the RI's RSA public key in octets.

After receiving C, the RI splits it into C_1 and C_2 and decrypts C_1 using its private key (consisting of a private exponent d and the modulus m), yielding Z:

C_1 | C_2 = C

c_1 = OS2IP(C_1, mLen)

Z = RSA.DECRYPT($PrivKey_{RI}$, c_1) = c_1^d mod m

where OS2IP converts an octet string to a nonnegative integer and is defined in [PKCS-1].

Using Z, the RI can derive KEK, and from KEK unwrap C_2 to yield K_{MAC} and $PREK_1$, …, $PREK_n$.

KEK = KDF(I2OSP(Z, mLen), **NULL**, kekLen)

K_{MAC} | $PREK_1$ | … | $PREK_n$ = AES-UNWRAP(KEK, C_2)

The following URI SHALL be used to identify this key transport scheme in <xenc:EncryptionMethod> elements:

http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-1#rsaes-kem-kdf2-kw-aes128

When creating corresponding RO for the recipient Device, the RI SHALL use PREK as the RO Encryption Key to form the <encKey> element of <roPayload> in subsequent RO Acquisition protocol. Note that although PREK (as an AES-WRAP output) is 192 bits, rather than 128 bits as is the (randomly generated) K_{REK} that is used to form the <encKey> element in [DRM-DRM-v2.1], this does not cause a backwards-compatibility problem, in that recipient Devices relative to the Move <userDomain>-constrained RO via RI protocol are SCE DRM Agents.

The recipient Device SHALL check whether the ID of the RI that issued the RO is present in the <moveIndication> element. If not the Device MUST reject the RO.

The recipient Device uses the relevant UDK to unwrap PREK. If the unwrapping fails the recipient Device MUST reject the RO. It then calculates Hash16 using the <moveIndication> element to recover K_{REK}.

14.Security Considerations

14.1 Replay Protection of Stateless Rights Object (Normative)

The replay cache specified in [DRM-DRM-v2.1] is only applied to stateful ROs. The same replay protection mechanism SHALL be applied to stateless ROs with <move> permission if acquired by an SCE DRM Agent through a method other than 2-pass RO Acquisition or an A2A protocol transaction or operation [SCE-A2A]..

15.Replay Cache Management

This section describes how a RI manages replay cache to prevent replay attack from malicious third party and to handle retry of request message from Source Device. The DRM Agent and RI MUST also support Replay Cache management mechanism defined in [DRM-DRM-v2.1].

When a RI receives a request message for Move Device RO via RI protocol or Move <userDomain>-constrained RO protocol, it SHALL check the <reqID> and the <nonce> element in the request message as follows:

- If the <reqID> element and <nonce> element in the request message matches with one of replay cache entry, the RI SHALL check the <time> element in the request message as follows:
    - If the <time> element in the request message matches with a <time> in the replay cache entry, the RI SHALL ignore the request message.
    - Else, the RI SHALL generate a response message which contains parameters from response information in the matching replay cache entry, and send the generated response message, and then abort the process. The RI SHALL NOT regenerate RO for Recipient Device nor send RO to the Recipient Device.
- Else, the RI SHALL create one replay cache entry which is at least composed of <reqID>, <nonce> and <time> which are copied from the request message, and continue to process the request message according to processing rules of each specific protocol (see section 9.1.2and 9.2.2). During the processing, whenever the RI generates the response message, the RI SHALL store a response information additionally into the created replay cache entry where the response information contains status, errorMessage, errorRedirectURL attributes and any relevant associated data in the response message.

It is strongly RECOMMENDED that the RI does not remove the replay cache entry until when the RI deems that the replay cache entry is old enough so that the source Device will no longer retry with the same nonce. In any case, the RI MUST keep the replay cache entry until the time when the RI would reject the request based on expiration, where the validity time window is RI implementation specific.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-SCE_DRM-V1_0-20110705-A	05 Jul 2011	Status changed to Approved by TP: OMA-TP-2011-0233-INP_SCE_V1_0_ERP_for_Final_Approval

Appendix B. Static Conformance Requirements (Normative)

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

B.1 SCR for DRM Agent

This section enumerates conformance requirements for DRM Agent. Since the following table extends Appendix G.1 Client Conformance Requirements in [DRM-DRM-v2.1], the SCE conformant DRM Agent SHALL also be conformant to OMA DRM 2.1.

Item	Function	Reference	Requirement
SCE-DRM-DRMAGENT-C-001-O	ROAP initiation from ROAP trigger message	6.1, 6.2, 6.3	
SCE-DRM-DRMAGENT-C-002-O	RO Acquisition protocol for User Domain RO	5.3, 8.1	SCE-DRM-DRMAGENT-C-003-O AND SCE-DRM-DRMAGENT-C-004-O
SCE-DRM-DRMAGENT-C-003-O	Support User Domain Authorization extension in ROAcquisitionRequest message	8.1.1	
SCE-DRM-DRMAGENT-C-004-O	Protected RO with <userDomain> constraint	8.1.2	
SCE-DRM-DRMAGENT-C-005-O	Move Device RO protocol	5.4.1, 8.2, 9.1	
SCE-DRM-DRMAGENT-C-006-O	Move <userDomain>-constrained RO protocol	5.4.2, 8.3, 9.2	
SCE-DRM-DRMAGENT-C-007-O	RO Upgrade protocol	5.5, 8.4, 9.3	
SCE-DRM-DRMAGENT-C-008-O	Support BatchRIURL header in DCF	6.4	

B.2 SCR for Rights Issuer

This section enumerates conformance requirements for Rights Issuer. Since the following table extends Appendix G.2 Server Conformance Requirements in [DRM-DRM-v2.1], the SCE conformant Rights Issuer SHALL also be conformant to OMA DRM 2.1.

Item	Function	Reference	Requirement
SCE-DRM-RI-S-001-O	ROAP trigger generation	6.1, 6.2, 6.3	
SCE-DRM-RI-S-002-O	RO Acquisition protocol for User Domain RO	5.3, 8.1	SCE-DRM-RI-S-005-O

Item	Function	Reference	Requirement
	without <userDomain> constraint		
SCE-DRM-RI-S-003-O	RO Acquisition protocol for User Domain with <userDomain> constraint	5.3, 8.1	SCE-DRM-RI-S-004-O AND SCE-DRM-RI-S-006-O
SCE-DRM-RI-S-004-O	Verification of User Domain Authorization extension in ROAcquisitionRequest message	8.1.1	
SCE-DRM-RI-S-005-O	Generation of Protected RO without <userDomain> constraint	8.1.2	
SCE-DRM-RI-S-006-O	Generation of Protected RO with <userDomain> constraint	8.1.2	
SCE-DRM-RI-S-007-O	Move Device RO protocol	5.4.1, 8.2, 9.1	
SCE-DRM-RI-S-008-O	Move <userDomain>-constrained RO protocol	5.4.2, 8.3, 9.2	
SCE-DRM-RI-S-009-O	RO Upgrade protocol	5.5, 8.4, 9.3	
SCE-DRM-RI-S-010-O	Support BatchRIURL header in DCF	6.4	

B.3 SCR for LRM

This section enumerates conformance requirements for LRM with at least Device key purpose (**`oma-kp-localRightsManagerDevice`**).

Item	Function	Reference	Requirement
SCE-DRM-LRMDEV-S-001-O	ROAP trigger generation	[DRM-DRM-v2.1]	
SCE-DRM- LRMDEV-S-002-O	RO Acquisition protocol for Device RO	[DRM-DRM-v2.1]	SCE-DRM- LRMDEV/RI-S-003-O
SCE-DRM- LRMDEV-S-003-O	Generation of Device bound Protected RO	8.1.2, [SCE-LRM]	
SCE-DRM-LRMDEV-	Generation of Protected RO with	8.1.2, [SCE-	SCE-DRM- LRMDEV-S-003-O

Item	Function	Reference	Requirement
S-004-O	<moveIndication> element	LRM]	

B.4 SCR for LRM

This section enumerates conformance requirements for LRM with at least Domain key purpose (**`oma-kp-localRightsManagerDomain`**).

Item	Function	Reference	Requirement
SCE-DRM-LRMDOM-S-001-O	ROAP trigger generation	[DRM-DRM-v2.1], 6.1, 6.2, 6.3	
SCE-DRM-LRMDOM-S-002-O	RO Acquisition protocol for User Domain RO	5.3, 8.1	SCE-DRM-LRMDOM/RI-S-003-O AND SCE-DRM-LRMDOM/RI-S-004-O
SCE-DRM-LRMDOM-S-003-O	Verification of User Domain Authorization extension in ROAcquisitionRequest message	8.1.1	
SCE-DRM-LRMDOM-S-004-O	Generation of User Domain RO	8.1.2, [SCE-LRM]	
SCE-DRM-LRMDOM-S-005-O	Generation of Protected RO with <moveIndication> element	8.1.2, [SCE-LRM]	SCE-DRM- LRMDOM-S-004-O

Appendix C. Example (Informative)

C.1 ROAP Message Examples

C.1.1 MoveDeviceRO Trigger

```
<?xml version="1.0" encoding="UTF-8"?>
<gen:drmTrigger
 type="moveDeviceRO"
 version="1.0"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:gen="urn:oma:xml:sce:gen"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:sceroap="urn:oma:drm:sce:roap"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <body id="idvalue0">
  <resID>
   <keyIdentifier xsi:type="gen:X509SPKIHash">
    <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
   </keyIdentifier>
  </resID>
  <resAlias>resAlias</resAlias>
  <nonce>c2FtcGxlVHJpZ2dlck5vbmNl</nonce>
  <reqURL>http://tempriuri.org</reqURL>
  <trgInfo>
   <sceroap:moveLRMRightsTrgInformation roRequested="true">
    <sceroap:recipientInfo>information string for recipientInfo</sceroap:recipientInfo>
   </sceroap:moveLRMRightsTrgInformation>
  </trgInfo>
 </body>
</gen:drmTrigger>
```

C.1.2 MoveDeviceRORequest

```
<?xml version="1.0" encoding="UTF-8"?>
<sceroap:moveDeviceRORequest
 triggerNonce="c2FtcGxlVHJpZ2dlck5vbmNl"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:gen="urn:oma:xml:sce:gen"
 xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
 xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
 xmlns:sceroap="urn:oma:drm:sce:roap"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <reqID>
  <keyIdentifier xsi:type="gen:X509SPKIHash">
   <hash>c2FtcGxlRFJNQWdlbnQ=</hash>
  </keyIdentifier>
 </reqID>
 <resID>
  <keyIdentifier xsi:type="gen:X509SPKIHash">
   <hash>c2FtcGxlUmlnaHRzSXNzdWVy</hash>
  </keyIdentifier>
 </resID>
 <nonce>RFJNQWdlbnROb25jZTA=</nonce>
```

```
  <time>2001-12-31T12:00:00</time>
  <reqInfo>
    <sceroap:moveDeviceRORequestInformation>
      <rcptDevID>
        <keyIdentifier xsi:type="gen:X509SPKIHash">
          <hash>c2FtcGxlUmVjaXBpZW50RGV2</hash>
        </keyIdentifier>
      </rcptDevID>
      <rightsInfo>
        <roID>n8yu98hy0e2109eu09ewf09u</roID>
        <rights></rights>
        <signature>...</signature>
        <stateInfo></stateInfo>
        <sourceDeviceID>
          <keyIdentifier xsi:type="gen:X509SPKIHash">
            <hash>c2FtcGxlRFJNQWdlbnQ=</hash>
          </keyIdentifier>
        </sourceDeviceID>
        <encKey Id="K_MAC_and_K_REK">
          <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128" />
          <ds:KeyInfo>
            <roap:keyIdentifier xsi:type="gen:X509SPKIHash">
              <hash>c2FtcGxlUmlnaHRzSXNzdWVy</hash>
            </roap:keyIdentifier>
          </ds:KeyInfo>
          <xenc:CipherData>
            <xenc:CipherValue>32fdsorew9ufdsoi09ufdskrew9urew0uderty5346wq</xenc:CipherValue>
          </xenc:CipherData>
        </encKey>
        <mac>j6lwx3rvEPO0vKtMup4NbeVu8nk=</mac>
      </rightsInfo>
    </sceroap:moveDeviceRORequestInformation>
  </reqInfo>
  <signature>321ue3ue3ue10ue2109ue1ueoidwoijdwe309u09ueqijdwqijdwq09uwqwqi009</signature>
</sceroap:moveDeviceRORequest>
```

C.1.3 MoveDeviceROResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<sceroap:moveDeviceROResponse
  status="Success"
  xmlns:gen="urn:oma:xml:sce:gen"
  xmlns:sceroap="urn:oma:drm:sce:roap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <reqID>
    <keyIdentifier xsi:type="gen:X509SPKIHash">
      <hash>c2FtcGxlRFJNQWdlbnQ=</hash>
    </keyIdentifier>
  </reqID>
  <resID>
    <keyIdentifier xsi:type="gen:X509SPKIHash">
      <hash>c2FtcGxlUmlnaHRzSXNzdWVy</hash>
    </keyIdentifier>
  </resID>
  <nonce>UmlnaHRzSXNzdWVyTm9uY2Uw</nonce>
```

```
  <resInfo>
    <sceroap:moveDeviceROResponseInformation>
      <prURL>http://sampleRI.org/nextURL</prURL>
    </sceroap:moveDeviceROResponseInformation>
  </resInfo>
  <signature>321ue3ue3ue10ue2109ue1ueoidwoijdwe309u09ueqijdwqijdwq09uwqwqi010</signature>
</sceroap:moveDeviceROResponse>
```

C.1.4 MoveUserDomainConstrainedRO Trigger

```
<?xml version="1.0" encoding="UTF-8"?>
<gen:drmTrigger
  type="moveUserDomainConstrainedRO"
  version="1.0"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:gen="urn:oma:xml:sce:gen"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:sceroap="urn:oma:drm:sce:roap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <body id="idvalue0">
    <resID>
      <keyIdentifier xsi:type="gen:X509SPKIHash">
        <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
      </keyIdentifier>
    </resID>
    <resAlias>resAlias</resAlias>
    <nonce>c2FtcGxlVHJpZ2dlck5vbmNl</nonce>
    <reqURL>http://tempriuri.org</reqURL>
    <trgInfo>
      <sceroap:moveUserDomainConstrainedROTrgInformation>
        <recipientInfo>information string for recipientInfo</recipientInfo>
      </sceroap:moveUserDomainConstrainedROTrgInformation>
    </trgInfo>
  </body>
</gen:drmTrigger>
```

C.1.5 MoveUserDomainConstrainedRORequest

```
<?xml version="1.0" encoding="UTF-8"?>
<sceroap:moveUserDomainConstrainedRORequest
  triggerNonce="c2FtcGxlVHJpZ2dlck5vbmNl"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:gen="urn:oma:xml:sce:gen"
  xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
  xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
  xmlns:sceroap="urn:oma:drm:sce:roap"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <reqID>
    <keyIdentifier xsi:type="gen:X509SPKIHash">
      <hash>c2FtcGxlRFJNQWdlbnQ=</hash>
    </keyIdentifier>
  </reqID>
  <resID>
    <keyIdentifier xsi:type="gen:X509SPKIHash">
      <hash>c2FtcGxlUmlnaHRzSXNzdWVy</hash>
```

```
    </keyIdentifier>
  </resID>
  <nonce>RFJNQWdlbnROb25jZTA=</nonce>
  <time>2001-12-31T12:00:00</time>
  <reqInfo>
    <sceroap:moveUserDomainConstrainedRORequestInformation>
     <userDomainConstrainedROInfo>…</userDomainConstrainedROInfo>
      <enc_REK>…</enc_REK>
      <sourceDeviceUserDomainAuthorization>…</sourceDeviceUserDomainAuthorization>
      <recipientDeviceId>
        <keyIdentifier xsi:type="gen:X509SPKIHash">
          <hash>c2FtcGxlUmVjaXBpZW50RGV2</hash>
        </keyIdentifier>
      </recipientDeviceId>
     <mac>…</mac>
    </sceroap:moveUserDomainConstrainedRORequestInformation>
  </reqInfo>
  <signature>321ue3ue3ue10ue2109ue1ueoidwoijdwe309u09ueqijdwqijdwq09uwqwqi009</signature>
</sceroap:moveUserDomainConstrainedRORequest>
```

C.1.6 MoveUserDomainConstrainedROResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<sceroap:moveUserDomainConstrainedROResponse
  status="Success"
  xmlns:gen="urn:oma:xml:sce:gen"
  xmlns:sceroap="urn:oma:drm:sce:roap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <reqID>
    <keyIdentifier xsi:type="gen:X509SPKIHash">
      <hash>c2FtcGxlRFJNQWdlbnQ=</hash>
    </keyIdentifier>
  </reqID>
  <resID>
    <keyIdentifier xsi:type="gen:X509SPKIHash">
      <hash>c2FtcGxlUmlnaHRzSXNzdWVy</hash>
    </keyIdentifier>
  </resID>
  <nonce>UmlnaHRzSXNzdWVyTm9uY2Uw</nonce>
  <resInfo>
    <sceroap:moveUserDomainConstrainedROResponseInformation>
      <RIUserDomainAuthorizationsuccess/>…</RIUserDomainAuthorization>
    </sceroap:moveUserDomainConstrainedROResponseInformation>
  </resInfo>
  <signature>321ue3ue3ue10ue2109ue1ueoidwoijdwe309u09ueqijdwqijdwq09uwqwqi010</signature>
</sceroap:moveUserDomainConstrainedROResponse>
```

C.1.7 ROUpgrade Trigger

```
<?xml version="1.0" encoding="UTF-8"?>
<gen:drmTrigger
  type="roUpgrade"
  version="1.0"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:gen="urn:oma:xml:sce:gen"
```

```
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:sceroap="urn:oma:drm:sce:roap"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <body id="idvalue0">
  <resID>
   <keyIdentifier xsi:type="gen:X509SPKIHash">
    <hash>aXENc+Um/9/NvmYKiHDLaErK0fk=</hash>
   </keyIdentifier>
  </resID>
  <resAlias>resAlias</resAlias>
  <nonce>c2FtcGxlVHJpZ2dlck5vbmNl</nonce>
  <reqURL>http://tempriuri.org</reqURL>
  <trgInfo>
   <sceroap:roUpgradeTrgInformation roRequested="true">
    <roID>roID</roID>
    <roAlias>roAlias</roAlias>
    <upgradeInfo>...</upgradeInfo>
   </sceroap:roUpgradeTrgInformation>
  </trgInfo>
 </body>
</gen:drmTrigger>
```

C.1.8 ROUpgradeRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<sceroap:roUpgradeRequest
 triggerNonce="c2FtcGxlVHJpZ2dlck5vbmNl"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:gen="urn:oma:xml:sce:gen"
 xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
 xmlns:roap="urn:oma:bac:dldrm:roap-1.0"
 xmlns:sceroap="urn:oma:drm:sce:roap"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <reqID>
  <keyIdentifier xsi:type="gen:X509SPKIHash">
   <hash>c2FtcGxlRFJNQWdlbnQ=</hash>
  </keyIdentifier>
 </reqID>
 <resID>
  <keyIdentifier xsi:type="gen:X509SPKIHash">
   <hash>c2FtcGxlUmlnaHRzSXNzdWVy</hash>
  </keyIdentifier>
 </resID>
 <nonce>RFJNQWdlbnROb25jZTA=</nonce>
 <time>2001-12-31T12:00:00</time>
 <reqInfo>
  <sceroap:roUpgradeRequestInformation>
   <existingRights>
    <roID>n8yu98hy0e2109eu09ewf09u</roID>
    <rights></rights>
    <signature>...</signature>
    <stateInfo></stateInfo>
    <sourceDeviceID>
     <keyIdentifier xsi:type="gen:X509SPKIHash">
      <hash>c2FtcGxlRFJNQWdlbnQ=</hash>
```

```
        </keyIdentifier>
      </sourceDeviceID>
      <encKey Id="K_MAC_and_K_REK">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128" />
        <ds:KeyInfo>
          <roap:keyIdentifier xsi:type="gen:X509SPKIHash">
            <hash>c2FtcGxlUmlnaHRzSXNzdWVy</hash>
          </roap:keyIdentifier>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>32fdsorew9ufdsoi09ufdskrew9urew0uderty5346wq</xenc:CipherValue>
        </xenc:CipherData>
      </encKey>
      <mac>j6lwx3rvEPO0vKtMup4NbeVu8nk=</mac>
    </existingRights>
    <ROUpgradeInfo>...</ROUpgradeInfo>
  </sceroap:roUpgradeRequestInformation>
 </reqInfo>
 <signature>321ue3ue3ue10ue2109ue1ueoidwoijdwe309u09ueqijdwqijdwq09uwqwqi009</signature>
</sceroap:roUpgradeRequest>
```

C.1.9 ROUpgradeResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<sceroap:roUpgradeResponse
 status="Success"
 xmlns:gen="urn:oma:xml:sce:gen"
 xmlns:sceroap="urn:oma:drm:sce:roap"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <reqID>
  <keyIdentifier xsi:type="gen:X509SPKIHash">
   <hash>c2FtcGxlRFJNQWdlbnQ=</hash>
  </keyIdentifier>
 </reqID>
 <resID>
  <keyIdentifier xsi:type="gen:X509SPKIHash">
   <hash>c2FtcGxlUmlnaHRzSXNzdWVy</hash>
  </keyIdentifier>
 </resID>
 <nonce>UmlnaHRzSXNzdWVyTm9uY2Uw</nonce>
 <resInfo>
  <sceroap:roUpgradeResponseInformation>
   <upgradeResult>
    <existingROID>ROID0000001</existingROID>
    <newRO>...</newRO>
   </upgradeResult>
  </sceroap:roUpgradeResponseInformation>
 </resInfo>
 <signature>321ue3ue3ue10ue2109ue1ueoidwoijdwe309u09ueqijdwqijdwq09uwqwqi010</signature>
</sceroap:roUpgradeResponse>
```