[MS-RMPR]:
Rights Management Services (RMS):
Client-to-Server Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
12/18/2006	0.1		MCPP Milestone 2 Initial Availability
03/02/2007	1.0		MCPP Milestone 2
04/03/2007	1.1		Monthly release
05/11/2007	1.2		Monthly release

Date	Revision History	Revision Class	Comments
06/01/2007	1.2.1	Editorial	Revised and edited the technical content.
07/03/2007	1.3	Minor	Updated technical content.
07/20/2007	1.3.1	Editorial	Revised and edited the technical content.
08/10/2007	1.3.2	Editorial	Revised and edited the technical content.
09/28/2007	1.3.3	Editorial	Revised and edited the technical content.
10/23/2007	1.4	Minor	Updated the technical content.
11/30/2007	1.4.1	Editorial	Revised and edited the technical content.
01/25/2008	1.4.2	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

The RMS: Client-to-Server Protocol is used to obtain and issue **certificates** and **licenses** used for creating and working with **protected content**. The RMS: Client-to-Server Protocol is a SOAP protocol. It consists of seven separate interfaces:

- Activation Service
- Certification Service
- Licensing Service
- Template Distribution Service
- Publishing Service
- Server Service
- Enrollment Cloud Service

The RMS: Client-to-Server Protocol depends on the proper use of these interfaces. In the case of the RMS 1.0 client, all five interfaces are used. RMS 1.0 SP1, RMS 1.0 SP2, and RMS 2.0 clients use all but the Activation Service. This specification contains the proper use of all five interfaces.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Active Directory (AD)
Advanced Encryption Standard (AES)
ASCII
Certificate Chain
Certification Authority (CA)
Coordinated Universal Time (UTC)
Data Encryption Standard (DES)
Domain
Domain Controller (DC)
Fully Qualified Domain Name (FQDN)
Globally Unique Identifier (GUID)
Little Endian
NT LAN Manager Protocol (NTLM)
Policy
Security Identifier (SID)
Unicode
Universal Naming Convention (UNC)

The following terms are specific to this document:

Certificate: As used in this document, **certificates** are expressed in XrML 1.2.

Client Licensor Certificate (CLC) Chain: An XrML 1.2 **certificate** chain that issues an asymmetric signing key pair to a user account, bound to a machine. The CLC grants the role of a user who can publish **protected content**.

Cloud Service: A set of one or more publicly available services that Microsoft operates.

Consumer: The user who uses **protected content**.

Endpoint: A network-specific address of a server process for remote procedure calls. The actual name of the **endpoint** depends on the RPC protocol sequence being used. For example, for the NCACN_IP_TCP RPC protocol sequence, an **endpoint** might be TCP port 1025. For more information, see [C706].

Hardware ID (HID): A unique string derived from a fingerprint of an individual computer. The **HID** is a unique identifier for the computer from which it is derived.

License: An XrML1.2 document that describes usage policy for **protected content**.

License Chain: Similar to a **certificate chain**, but for a **license**.

Offline Publishing: The process of creating **protected content** and signing the associated **publishing license** using a previously acquired CLC.

Online Publishing: The process of creating **protected content** and contacting a server to have the **publishing license** signed.

Protected Content: Any content or information (file, e-mail) that has an RMS usage policy assigned to it and is encrypted according to that policy. Also known as "Protected Information".

Publishing License (PL): An XrML 1.2 **license** that defines usage policy for **protected content** and contains the content key with which that content is encrypted. The usage policy identifies all authorized users and the actions they are authorized to take with the content, along with any conditions on that usage. The **publishing license** tells the server what usage policies apply to a given piece of content and grants the server the right to issue **use licenses (ULs)** based on that policy. The **PL** is created when content is protected. Also known as an "Issuance License (IL)".

Passport Unique ID (PUID): A unique user name associated with a Microsoft Passport account.

Rights Policy Template: An XrML 1.2 document that contains a predefined usage policy that is used to create the **PL** when content is protected. Conceptually, a **rights policy template** (or "template") is a blueprint for a **PL**, identifying authorized users and the actions they are authorized to take with the content (along with any conditions on that usage). Unlike a **PL**, a **template** does not contain a content key or information about the content owner. The content key and information about the content owner are required to be added when the **PL** for a given piece is created from the template. End users can use a **template** when protecting a document instead of defining the specifics of the usage policy themselves. When a document is published using a **template**, the **template** is used to generate the **PL**.

RMS Account Certificate (RAC): An XrML 1.2 **certificate** that issues an asymmetric encryption key pair to a user account and binds that user account to a specific computer. The **RAC** grants the role of a user who can access **protected content**. Also known as a "Group Identity Credential (GIC)", the **RAC** exists as an XrML 1.2 **certificate chain** that issues an asymmetric encryption key pair to a user account, bound to a machine.

Security Processor: A trusted component on the client machine that enforces usage policy. It has exclusive access to the **Security Processor Certificate (SPC)** private key.

Security Processor Certificate (SPC): An XrML 1.2 **certificate chain** generated during activation that contains the public key corresponding to the **SPC** private key. The **SPC** grants the role of a machine that can be used for working with **protected content**.

Security Processor Certificate (SPC) Private Key: A unique private key that is generated at activation time and issued to the machine, either by self-activation or by calling the **Activate** method. Also known as a "Machine Key".

Server Licensor Certificate (SLC): An XrML 1.2 **certificate** that issues an asymmetric key pair to a server for encryption and signing. The **SLC** grants the role of a server that can issue a **certificate** and **license** for working with **protected content**.

Use License (UL): An XrML 1.2 **license** that authorizes a user to access a given **protected content** file and describes the usage policies that apply. Also known as an "End-User License (EUL)".

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-MWBE] Microsoft Corporation, "Microsoft Web Browser Federated Sign-On Protocol Extensions", January 2007.

[MS-MWBF] Microsoft Corporation, "Microsoft Web Browser Federated Sign-On Protocol Specification", January 2007.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2396] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SOAP1.2/1] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624

[SOAP1.2/2] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[WSDLExt] Nielsen, H.F., Christensen, E., and Farrell, J., "WS-Attachments", June 2002, http://xml.coverpages.org/draft-nielsen-dime-soap-01.txt

If you have any trouble finding [WSDLExt], please check here.

[XML10] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Third Edition)", February 2004, http://www.w3.org/TR/REC-xml

[XMLNS] World Wide Web Consortium, "Namespaces in XML 1.0 (Second Edition)", August 2006, http://www.w3.org/TR/REC-xml-names/

[XMLSCHEMA1] Thompson, H.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[XPATH] Clark, J. and DeRose, S., "XML Path Language (XPath), Version 1.0", W3C Recommendation, November 1999, http://www.w3.org/TR/xpath

[XRML] ContentGuard, Inc., "XrML... eXtensible rights Markup Language", 2005, http://www.xrml.org/XrML_12.asp

If you have any trouble finding [XRML], please check here.

1.2.2 Informative References

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification", June 2007.

1.3 Protocol Overview (Synopsis)

The RMS: Client-to-Server Protocol provides support for information protection through content encryption and fine-grained **policy** definition and enforcement. In doing so, the RMS: Client-to-Server Protocol enables end users to create and access protected information. This specification defines the RMS: Client-to-Server Protocol, which is a SOAP-based protocol that uses HTTP 1.1 as its transport.

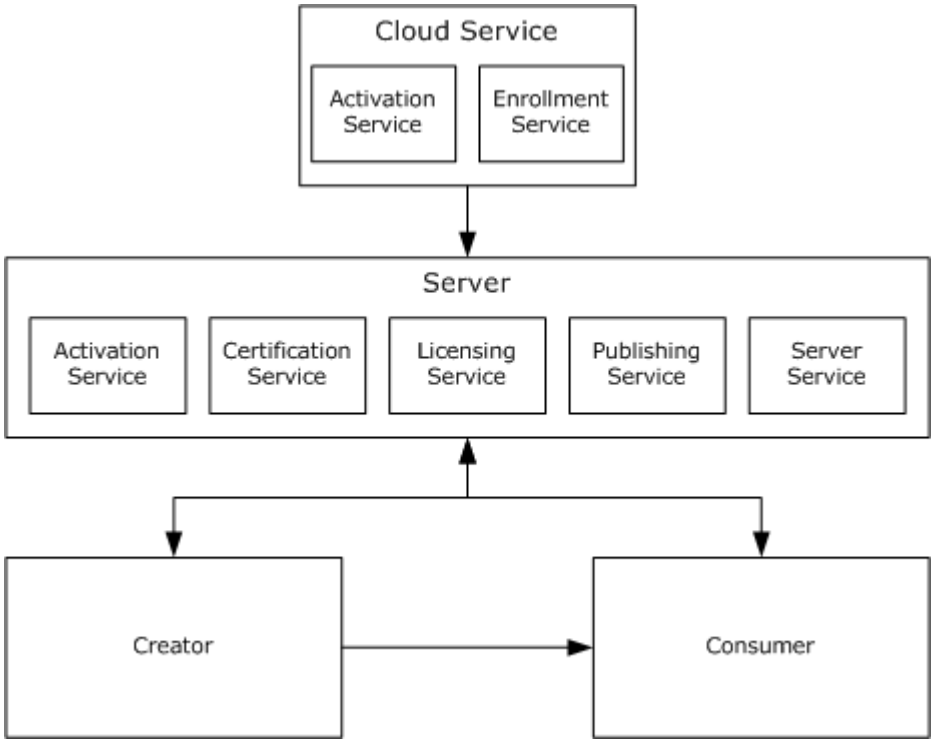

Figure 1: Rights management roles

The RMS system involves four active entities: the creator, the **consumer**, the server, and the **cloud service**.

The server is required to undergo a one-time bootstrapping process to begin functioning in the RMS system. In RMS 1.0, RMS 1.0 SP1, and RMS 1.0 SP2, this operation involves contacting the cloud service. In RMS 2.0, this operation is done entirely offline. The creator and consumer contact the server for a one-time bootstrapping process to functioning in the RMS system.

The creator builds a document and chooses an access policy for that document, either by creating it directly or by using a **rights policy template** to apply a predefined access policy. The creator then encrypts the document and binds the access policy to that document in the form of a **PL**.

The consumer, upon receiving the document from the creator and opening it, supplies the server with the PL and the **RMS account certificate (RAC)** that was acquired during bootstrapping. If the consumer is allowed access according to the access policy in the PL, the server issues the consumer a **use license (UL)** that specifies the access policy for the consumer and binds the content decryption key to the consumer's RAC. The RAC key is encrypted by the key of a trusted software module called the **security processor**. When the consumer attempts to access the document, the security processor decides whether the requesting application on the consumer machine is capable of enforcing the access policy. If so, it supplies plain text of the document to the application along with the policy that the application is to enforce.

A client can play the role of a creator, a consumer, or both, depending on implementation. The client is responsible for requesting certificates, licenses, and policies from the server. It is further responsible for enforcing authorization policies as they apply to protected information and encrypting or decrypting content as appropriate. The RMS 2.0 client in Windows Vista SP1 and Windows Server 2008 can also fetch rights policy templates from an RMS 2.0 server.

The server role in the RMS: Client-to-Server Protocol is responsible for issuing certifications, keys, and authorization policies, and for signing these issued certificates and policies with keys it holds in escrow. It is further responsible for evaluating and issuing authorization policies based on identity credentials the client provides in protocol requests.

The RMS: Client-to-Server Protocol consists of a number of service **endpoints**, and each endpoint provides one or more remote procedures that are related in function to each other. The Web server implementation identifies and services the endpoints, and the Web server describes the endpoint's interface using the Web Services Description Language ([WSDL]), which is analogous to a COM IDL.

The remote procedures are invoked to:

- Acquire or exchange certificates.
- Request an authorization policy for protected information.
- Author an authorization policy for protected information.
- Discover information about the server or a user that is necessary for client operation.
- Manage the server remotely.

The RMS: Client-to-Server Protocol is stateless and the methods on the protocol can generally be called in any order. The following illustration shows the message exchange.

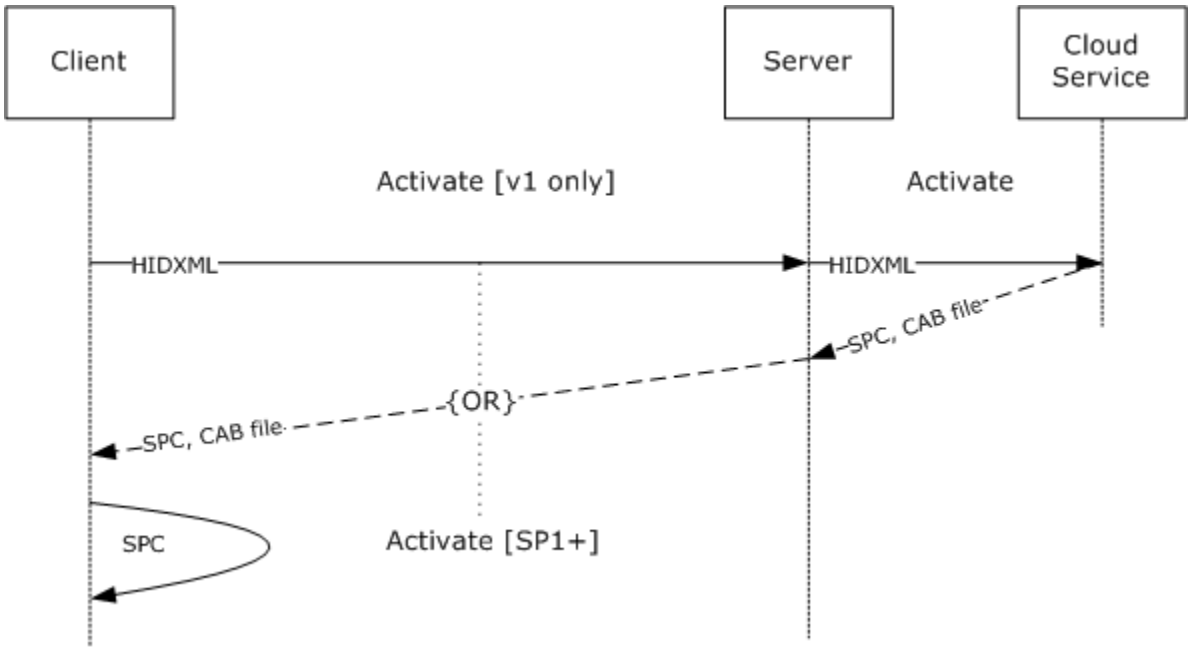

Figure 2: Client/server message exchange

1.3.1 Server Bootstrapping

Server bootstrapping is an initialization step that the server completes before it services any client requests. During server bootstrapping, the server generates its key pair and builds an unsigned **server licensor certificate (SLC)** that contains its public key. Before the client trusts the SLC, it has to be signed by the Microsoft RMS enrollment cloud service. The server makes an enroll request

to the Microsoft RMS enrollment cloud service by submitting its unsigned SLC, and the server returns a signed SLC chain that leads back to the shared RMS root key.

1.3.2 Client Bootstrapping

Client bootstrapping is a set of initialization steps that clients complete before moving on to either **offline publishing** or licensing. Client bootstrapping is not a prerequisite for **online publishing**.

Client bootstrapping involves the following request and response methods: Activate, Certify, FindServiceLocationsForUser, and GetClientLicensorCert.

1.3.3 Online Publishing

Online publishing does not require completion of the client bootstrapping steps. When the client is used to protect content, it generates a PL that contains the usage policy and the content key. The PL encrypts the usage policy and the content key to a server and identifies the server that can be used to issue ULs from the PL. During online publishing, the client acquires the SLC of the server in order to encrypt the usage policy and content key to the server and build the PL chain.

The following request and response methods are used for online publishing: GetLicensorCertificate and AcquireIssuanceLicense.

1.3.4 Template Acquisition

The RMS 2.0 client in Windows Vista SP1 and Windows Server 2008 can acquire rights policy templates from an RMS 2.0 server. The client makes an AcquireTemplateInformation request to the server. The server returns information about the available templates. The client makes a subsequent AcquireTemplates request to the server for outdated and missing templates, deleting templates that are no longer present on the server from its local license store. The client then places the newly obtained templates from the server in its local license store.

The following request and response methods are used for template acquisition: AcquireTemplateInformation and AcquireTemplates.

1.3.5 Offline Publishing

Offline publishing does not make a call to the server. The client is required to have a valid **client licensor certificate (CLC) chain**, RAC, and **security processor certificate (SPC)** to publish offline. For an overview of the bootstrapping process, see sections 1.3.1 and 1.3.2.

When the client is used to protect content, it generates a PL that contains the usage policy and the content key. The PL encrypts the usage policy and the content key to a server and identifies the server that can be used to issue use licenses from the PL.

During offline publishing, the usage policy and content key are encrypted using the server's public key from the issuer of the CLC. The PL is signed using the CLC private key, and the resultant signed PL chain includes the PL, CLC, and SLC from the CLC chain.

There are no request and response methods used for offline publishing.

1.3.6 Licensing

A UL is required for a user to access protected content. The UL describes the usage policies that apply to the user while accessing a particular protected content file. It also contains the content key encrypted with the user's RAC public key.

The client is required to possess a valid RAC and SPC to access protected content. For an overview of the bootstrapping process, see section 1.3.1. The client needs a valid PL to acquire a UL for protected content. For more information about publishing and PLs, see sections 1.3.3 and 1.3.5.

The following request and response method is used for licensing: AcquireLicense.

1.4 Relationship to Other Protocols

The RMS: Client-to-Server Protocol uses the SOAP messaging protocol, as specified in [SOAP1.1], for formatting requests and responses. It transmits these messages using the HTTP and/or HTTPS protocols. SOAP is considered the wire format used for messaging, and HTTP and HTTPS are the underlying transport protocols. The content files are downloaded using HTTP 1.1, as specified in [RFC2616].

The RMS: Client-to-Server Protocol user certification endpoint uses authentication to determine the requesting user's identity.

The RMS: Client-to-Server Protocol can use the Microsoft Web Browser Federated Sign-On Protocol, as specified in [MS-MWBF], on requests to the licensing or user certification endpoints for providing user authentication. Its extensions are defined in the Microsoft Web Browser Federated Sign-on Protocol Extensions, as specified in [MS-MWBE].

The RMS: Client-to-Server Protocol is composed of Web services using SOAP [SOAP1.1] over HTTP or HTTPS [RFC2616], for communication.

The following diagram shows the transport stack that the RMS: Client-to-Server Protocol uses.

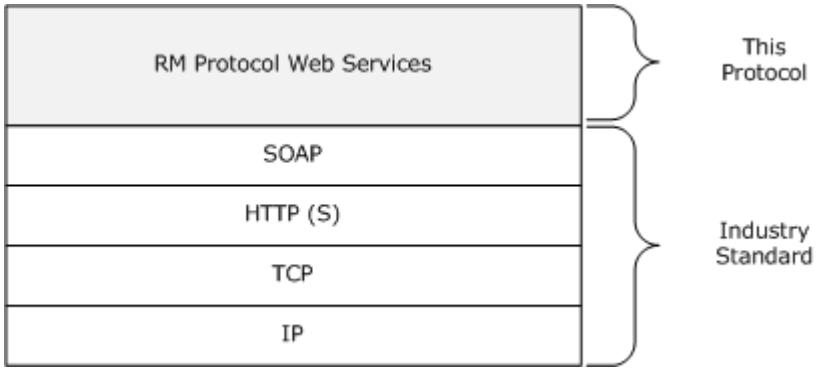

Figure 3: RMS: Client-to-Server Protocol transport stack

Content download is accomplished using HTTP 1.1 GET Byte Range requests, as specified in [RFC2616] section 14.35.

1.4.1 DRM Namespaces

The RMS: Client-to-Server Protocol defines the following Digital Rights Management (DRM) namespaces for use with the SOAP transport:

- http://microsoft.com/DRM/AdminService
- http://microsoft.com/DRM/CertificationService
- http://microsoft.com/DRM/EditIssuanceLicenseService

- http://microsoft.com/DRM/LicensingService
- http://microsoft.com/DRM/PublishingService

1.5 Prerequisites/Preconditions

The RMS: Client-to-Server Protocol assumes that the client is able to discover the server, either by being able to accessing the appropriate **Active Directory (AD)** object or by some other means.

With all versions of RMS, the protocol assumes that a client-side package is installed so that RMS-aware applications can perform RMS operations. The client-side package includes the RMS client APIs, and the code that underlies these APIs contains the algorithms and connection objects for creating RMS protocol requests and processing responses.

Finally, it is assumed that the protected information itself can be distributed in some way, as the RMS: Client-to-Server Protocol is not involved in content distribution.

1.6 Applicability Statement

The RMS: Client-to-Server Protocol is used to manage user access to protected information, such as documents, e-mail, and files. The RMS: Client-to-Server Protocol can be used in an AD configuration or as a stand-alone service provider. For more information on AD, see [MS-ADTS].

1.7 Versioning and Capability Negotiation

This specification covers versioning issues in the following areas:

- Supported Transports: This protocol is implemented on top of HTTP and SOAP, as specified in section 2.1.
- Protocol Versions: The RMS: Client-to-Server Protocol client has versions 1.0, 1.0 SP1, 1.0 SP2, and 2.0. The client version is transparent to the server.
- Security and Authentication Methods: The SOAP protocol passively supports **NT LAN Manager (NTLM)** authentication over HTTP or HTTPS, as specified in [NTLM].
- Localization: The RMS: Client-to-Server Protocol has no localization-dependent behaviors.
- Capability Negotiation: The RMS: Client-to-Server Protocol supports limited capability negotiation via the VersionData type that is present on all protocol requests. On a request, the VersionData structure contains a MinimumVersion and MaximumVersion value indicating the range of versions the client is capable of understanding. On a response, the VersionData structure contains a MinimumVersion and MaximumVersion that the server is capable of understanding.<1>

  This protocol can be spread across multiple servers. To determine which servers are capable of specific methods, the client calls the **FindServiceLocationsForUser (section 3.1.7.3)** method in the Server Service (section 2.2.7).

1.8 Vendor-Extensible Fields

This protocol does not contain any vendor-extensible fields. All XML schema are considered nonextensible in the RMS: Client-to-Server Protocol.

1.9 Standards Assignments

The RMS: Client-to-Server Protocol uses the following standard XML namespaces:

- http://schemas.xmlsoap.org/wsdl/http/
- http://www.w3.org/2001/XMLSchema
- http://schemas.xmlsoap.org/wsdl/soap/
- http://schemas.xmlsoap.org/wsdl/soap12/
- http://schemas.xmlsoap.org/soap/encoding/
- http://schemas.xmlsoap.org/wsdl/

2 Messages

2.1 Transport

A RMS: Client-to-Server Protocol message MUST be formatted as specified in either [SOAP1.1] or [SOAP1.2/1].<2>

Each RMS Web service MUST support SOAP [SOAP1.1] over HTTP [RFC2616] over TCP/IP. Each RMS Web service SHOULD support HTTPS for securing its communication with clients.<3> Each RMS Web service MUST require HTTPS for communication with clients when making a request enabled by the Microsoft Web Browser Federated Sign-on Protocol [MS-MWBF] to the Licensing or Certification Web services.

The URLs specified in 2.2.6 MUST be exposed by the server as endpoints for the HTTP and SOAP over HTTP transports.

To optimize network bandwidth, the client implementation MAY request the reply be compressed by specifying the encoding format in the HTTP Accept-Encoding request-header field as specified in [RFC2616], section 14.3. The update server SHOULD encode the reply using the requested format.

2.2 Common Data Types

2.2.1 Common Schema

The elements described in this section are present in multiple services.

2.2.1.1 Certificate Element

The **Certificate (ArrayOfXmlNode)** element encloses any eXtensible Rights Markup Language (as specified in [XrML]) certificate parameter that can be represented as a literal within an XML element on the protocol.

```
<xs:element name="Certificate">
  <xs:complexType
    mixed="true"
  >
    <xs:sequence>
      <xs:any
        namespace=""
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.2.1.2 CertificateChain Element

The **CertificateChain (LicensorCertChain)** element uses an array of XML elements to represent a certificate chain. This element MUST contain a valid certificate chain, as specified in 2.3.

```
<xs:element name="CertificateChain"
  type="ArrayOfXmlNode"
 />
```

2.2.1.3 VersionData Element

The **VersionData** element contains versioning information that serves as a declaration of the capability support necessary to understand and process the entire request or response.

```
<xs:element name="VersionData"
  type="VersionData"
 />
```

2.2.1.4 string Element

The **string (ArrayOfString)** element is an extra XML wrapper for the string data type. This element helps define the **string (ArrayOfString)** element as an array of ordinary XML strings. This element MUST contain only one literal string.

```
<xs:element name="string"
  type="string"
 />
```

2.2.1.5 MaximumVersion Element

The **MaximumVersion (VersionData)** element is used to specify the maximum capability version requirement of the RMS: Client-to-Server Protocol between client and server.

```
<xs:element name="MaximumVersion"
  type="string"
 />
```

2.2.1.6 MinimumVersion Element

The **MinimumVersion (VersionData)** element is used to specify the minimum capability version requirement of the RMS: Client-to-Server Protocol between client and server.

```
<xs:element name="MinimumVersion"
  type="string"
 />
```

2.2.1.7 URL Element

The **URL (ServiceLocationResponse)** element defines the use of the string data type to represent a URL in the RMS: Client-to-Server Protocol. This element MUST contain a literal string.

```
<xs:element name="URL"
```

```
    type="string"
  />
```

2.2.2 Common Complex Types

The complex types described in this section are present in multiple services.

2.2.2.1 ArrayOfXmlNode Complex Type

The **ArrayOfXmlNode** complex type contains an array of XML elements. It is used exclusively for exchanging XrML certificates, each of which MUST be represented as an XML fragment. Each XML fragment is enclosed in the **Certificate** element. For more information on XrML, see [XRML].

```
<xs:complexType name="ArrayOfXmlNode">
  <xs:sequence>
    <xs:element name="Certificate"
      minOccurs="0"
      maxOccurs="unbounded"
    >
      <xs:complexType
        mixed="true"
      >
        <xs:sequence>
          <xs:any
            namespace=""
           />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
Certificate	N/A

2.2.2.2 VersionData Complex Type

The **VersionData** complex type is used to represent the capability version of the client and server. The version data in this type MUST be represented by using a literal string and MUST conform to the format "a.b.c.d". Subversion value "a" MUST be the most major component of the version, value "b" MUST be the next most major, value "c" MUST be the next most major, and "d" MUST be the minor subversion value.

The client and server SHOULD<4> negotiate the capability version of the protocol to be used in client/server exchanges.

```
<xs:complexType name="VersionData">
  <xs:sequence>
```

```
      <xs:element name="MinimumVersion"
        type="string"
        minOccurs="0"
        maxOccurs="1"
       />
      <xs:element name="MaximumVersion"
        type="string"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
```

Child Elements

Element	Type
MinimumVersion	**string**
MaximumVersion	**string**

2.2.3 Activation Service Schema

The complex types, simple types, and elements that are described in this section are used in the Activation Service.

2.2.3.1 Activate Element

The **Activate** element contains the body of the message for the Activate Web method. The Activate Web method parameters consist of any number of **hardware IDs (HIDs)** that are associated with the Microsoft Activation Service.

```
<xs:element name="Activate">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="requestParams"
        type="ArrayOfActivateParams"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
requestParams	**ArrayOfActivateParams**

2.2.3.1.1 ArrayOfActivateParams Complex Type

The **ArrayOfActivateParams** complex type contains an array of parameters for the Activate request operation. This array consists of any number of **ActivateParams (section 2.2.3.1.1.1)**.

```
<xs:complexType name="ArrayOfActivateParams">
  <xs:sequence>
    <xs:element name="ActivateParams"
      type="ActivateParams"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
ActivateParams	**ActivateParams**

2.2.3.1.1.1 ActivateParams Complex Type

The **ActivateParams** complex type contains a single HID represented in XML form.

```
<xs:complexType name="ActivateParams">
  <xs:sequence>
    <xs:element name="HidXml"
      minOccurs="0"
      maxOccurs="1"
    >
      <xs:complexType name="XmlNode"
        mixed="true"
      >
        <xs:sequence>
          <xs:any
            namespace=""
           />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
HidXml	XmlNode

2.2.3.1.1.1.1 HidXml Element

The format and content of the HID is implementation-dependent.<5> The HID SHOULD uniquely identify the client making the Activate request. The server operates transparently on the HID, serving only as a pass-through to the Microsoft RMS Activation cloud service. The SOAP operations that the server and the cloud service use while the server is acting as a pass-through are identical to those made between the client and the server. For information on how to use the cloud service, see section 3.1.3.1.

```
<xs:element name="HidXml">
  <xs:complexType
    mixed="true"
  >
    <xs:sequence>
      <xs:any
        namespace=""
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.2.3.2 ActivateResponse Element

The **ActivateResponse** element contains the body of the response from the Activate method. The Activate method response consists of any number of BinarySignatures and MachineCertificateChains.

```
<xs:element name="ActivateResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ActivateResult"
        type="ArrayOfActivateResponse"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
ActivateResult	**ArrayOfActivateResponse**

2.2.3.2.1 ArrayOfActivateResponse Complex Type

The **ArrayOfActivateResponse** complex type contains an array of responses to an Activate request operation.

```
<xs:complexType name="ArrayOfActivateResponse">
  <xs:sequence>
```

```
    <xs:element name="ActivateResponse"
      type="ActivateResponse"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
ActivateResponse	**ActivateResponse**

2.2.3.2.1.1 ActivateResponse Complex Type

The **ActivateResponse** complex type contains an array of machine certificates and a binary signature to verify the binary data the server returns in a DIME attachment (as described in [WSDLExt]) on this Activate Web method response. The BinarySignature and attachment are passed through by the server and treated as transparent data.

```
<xs:complexType name="ActivateResponse">
  <xs:sequence>
    <xs:element name="BinarySignature"
      minOccurs="0"
      maxOccurs="1"
    >
      <xs:complexType
        mixed="true"
      >
        <xs:sequence>
          <xs:any
            namespace=""
           />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="MachineCertificateChain"
      type="ArrayOfXmlNode"
      maxOccurs="0"
      minOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
BinarySignature	N/A

Element	Type
MachineCertificateChain	ArrayOfXmlNode

2.2.3.2.1.1.1 BinarySignature Element

The **BinarySignature (ActivateResponse)** element is a fragment of XML that contains a signed hash of the binary data returned by the server in a DIME attachment (as described in [WSDLExt]) on the Activate Web method response. The **BinarySignature** and attachment are passed through by the server and treated as transparent data.

```
<xs:element name="BinarySignature">
  <xs:complexType
    mixed="true"
  >
    <xs:sequence>
      <xs:any
        namespace=""
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.2.4 Certification Service Schema

The complex types, simple types, and elements described in this section are used in the Certification Service.

2.2.4.1 Certify Element

The **Certify** element contains the body of the request for the Certify Web method.

```
<xs:element name="Certify">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="requestParams"
        type="CertifyParams"
        minOccurs="1"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
requestParams	CertifyParams

2.2.4.1.1 CertifyParams Complex Type

The **CertifyParams** complex type allows the Certify request operation to accept a list of machine certificates for performing the certificate operation. The list of machine certificates is stored in an array. The **ArrayOfXmlNode (section 2.2.2.1)** complex type serves as a wrapper for this array. The Persistent parameter is a Boolean flag that indicates whether the response is a temporary identity certificate with a short Time to Live (TTL) determined by the server (when the value is TRUE), or an identity certificate with a normal TTL (when the value is FALSE).

```
<xs:complexType name="CertifyParams">
  <xs:sequence>
    <xs:element name="MachineCertificateChain"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
    <xs:element name="Persistent"
      type="boolean"
      minOccurs="1"
      maxOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
MachineCertificateChain	**ArrayOfXmlNode**
Persistent	**boolean**

2.2.4.2 CertifyResponse Element

The **CertifyResponse** element contains the response to a Certify request operation. This element is used as an out parameter for the Certify operation.

```
<xs:element name="CertifyResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="CertifyResult"
        type="CertifyResponse"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
CertifyResult	**CertifyResponse**

2.2.4.2.1 CertifyResponse Complex Type

The **CertifyResponse** complex type contains response parameters consisting of an array of certificates and certificate quota data. The certificates represent the user identity certificate that the server issues. The quota data SHOULD NOT be used.

```
<xs:complexType name="CertifyResponse">
  <xs:sequence>
    <xs:element name="CertificateChain"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
    <xs:element name="Quota"
      type="QuotaResponse"
      minOccurs="0"
      maxOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
CertificateChain	**ArrayOfXmlNode**
Quota	**QuotaResponse**

2.2.4.2.1.1 QuotaResponse Complex Type

The server does not process the **QuotaResponse** complex type. The *Verified* parameter value MUST be set to true. The *CurrentConsumption* parameter value MUST be less than the *Maximum* parameter value, otherwise arbitrary values for these two parameters MAY<6> be used.

```
<xs:complexType name="QuotaResponse">
  <xs:sequence>
    <xs:element name="Verified"
      type="boolean"
      minOccurs="1"
      maxOccurs="1"
     />
    <xs:element name="CurrentConsumption"
      type="int"
      minOccurs="1"
      maxOccurs="1"
     />
    <xs:element name="Maximum"
      type="int"
```

```
        minOccurs="1"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
```

Child Elements

Element	Type
Verified	**boolean**
CurrentConsumption	**int**
Maximum	**int**

2.2.5 Licensing Service Schema

The complex types, simple types, and elements described in this section are used in the Licensing Service.

2.2.5.1 AcquireLicense Element

The **AcquireLicense** element contains the body of the request for the AcquireLicense Web method. The *RequestParams* parameter contains an array of any number of sets of **license chains** used for license acquisition.

```
<xs:element name="AcquireLicense">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="RequestParams"
        type="ArrayOfAcquireLicenseParams"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
RequestParams	**ArrayOfAcquireLicenseParams**

2.2.5.1.1 ArrayOfAcquireLicenseParams Complex Type

The **ArrayOfAcquireLicenseParams** complex type contains any number of sets of **AcquireLicenseParams** used to acquire a license.

```
<xs:complexType name="ArrayOfAcquireLicenseParams">
  <xs:sequence>
    <xs:element name="AcquireLicenseParams"
      type="AcquireLicenseParams"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
AcquireLicenseParams	AcquireLicenseParams

2.2.5.1.1.1 AcquireLicenseParams Complex Type

The **AcquireLicenseParams** complex type defines the parameters that are used to acquire a single license. *LicenseeCerts* is an **ArrayOfXmlNode** that represents certificates that the client provides to successfully complete the request. A user identity certificate, issued by way of the Certify method, MUST be presented in this parameter. The user identity MUST be signed by an issuer with which the server has a trust relationship.

IssuanceLicense is an **ArrayOfXmlNode** that represents the usage policy for the protected information. The usage policy MUST be signed by an issuer with which the server has a trust relationship.

ApplicationData is a fragment of XML that the client provides and that the server does not use; it is included verbatim in the certificate issued in response to the client. A client MAY specify a null value for this parameter.

The format of the certificates in this complex type are specified in section 2.3.

```
<xs:complexType name="AcquireLicenseParams">
  <xs:sequence>
    <xs:element name="LicenseeCerts"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
    <xs:element name="IssuanceLicense"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
    <xs:element name="ApplicationData"
      minOccurs="0"
      maxOccurs="1"
    >
      <xs:complexType
        mixed="true"
      >
        <xs:sequence>
```

```
            <xs:any
              namespace=""
             />
           </xs:sequence>
         </xs:complexType>
       </xs:element>
     </xs:sequence>
   </xs:complexType>
```

Child Elements

Element	Type
LicenseeCerts	ArrayOfXmlNode
IssuanceLicense	**ArrayOfXmlNode**
ApplicationData	N/A

2.2.5.1.1.1.1 ApplicationData Element

The **ApplicationData (AcquireLicenseParams)** element contains application data wrapped in an XML element. The client provides the **ApplicationData (AcquireLicenseParams)** element, which the server does not use but which is included verbatim in the certificate issued in response to the client. A client MAY specify a null value for this parameter.

```
<xs:element name="ApplicationData">
  <xs:complexType
    mixed="true"
  >
    <xs:sequence>
      <xs:any
        namespace=""
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.2.5.2 AcquireLicenseResponse Element

The **AcquireLicenseResponse** element contains the response to an AcquireLicense Web method request. The *AcquireLicenseResult* parameter is an array of **certificate chains** that contains a licensed certificate that corresponds to the original **AcquireLicense** (section 2.2.5.1) request.

```
<xs:element name="AcquireLicenseResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AcquireLicenseResult"
        type="ArrayOfAcquireLicenseResponse"
        minOccurs="0"
        maxOccurs="1"
```

```
        />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Child Elements

Element	Type
AcquireLicenseResult	ArrayOfAcquireLicenseResponse

2.2.5.2.1 ArrayOfAcquireLicenseResponse Complex Type

The **ArrayOfAcquireLicenseResponse** complex type contains any number of **AcquireLicenseResponse** elements.

```
<xs:complexType name="ArrayOfAcquireLicenseResponse">
  <xs:sequence>
    <xs:element name="AcquireLicenseResponse"
      type="AcquireLicenseResponse"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
AcquireLicenseResponse	AcquireLicenseResponse

2.2.5.2.1.1 AcquireLicenseResponse Complex Type

The **AcquireLicenseResponse** complex type defines the parameters returned from an **AcquireLicense** operation. A valid response MUST include a **CertificateChain (LicensorCertChain)** parameter that is an **ArrayOfXmlNode** that represents the authorization policy the server issues to the client. A **ReferenceCertificates** parameter is an **ArrayOfXmlNode** that represents other certificates, not part of the authorization policy, that the server returns to the client. The **ReferenceCertificates** response parameter SHOULD<7> be empty.

```
<xs:complexType name="AcquireLicenseResponse">
  <xs:sequence>
    <xs:element name="CertificateChain"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
    <xs:element name="ReferenceCertificates"
      type="ArrayOfXmlNode"
```

```
      minOccurs="0"
      maxOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
CertificateChain	**ArrayOfXmlNode**
ReferenceCertificates	**ArrayOfXmlNode**

2.2.5.3 AcquireTemplateInformation Element

The **AcquireTemplateInformation** element contains the body of the request for the **AcquireTemplateInformation** Web method.

```
<xs:element name="AcquireTemplateInformation">
  <xs:complexType />
</xs:element>
```

2.2.5.4 AcquireTemplateInformationResponse Element

The **AcquireTemplateInformationResponse** element contains the response to an **AcquireTemplateInformationResponse** Web method.

```
<xs:element name="AcquireTemplateInformationResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AcquireTemplateInformationResult"
        type="TemplateInformation"
        maxOccurs="1"
        minOccurs="0"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
AcquireTemplateInformationResult	**TemplateInformation**

2.2.5.4.1 TemplateInformation Complex Type

The **TemplateInformation** complex type contains any number of elements.

The **TemplateInformation** complex type defines the parameters returned from an **AcquireTemplateInformation** operation. A valid response MUST include one *ServerPublicKey* parameter. This parameter MUST be a string that represents the RSA PKCS#1-encoded public key (as specified in [PKCS1]) of the server's SLC, base64-encoded. This public key string SHOULD be used only to identify the server and SHOULD NOT be used for any cryptographic operations. The client SHOULD use this public key when comparing the set of templates it already has with those available from the server. The response MUST also include one *GuidHashCount* parameter that is an integer that represents the total number of **GuidHash** elements that are included in the response. The next parameter is *GuidHash*, which is of complex type **GuidHash**, and represents a **GUID** and hash pair for a template. The response contains a *GuidHash* parameter for all the templates available on the server. The number of **GuidHash** elements MAY range from "0" to "unlimited".

```
<xs:complexType name="TemplateInformation">
  <xs:sequence>
    <xs:element name="ServerPublicKey"
      type="String"
      maxOccurs="1"
      minOccurs="0"
     />
    <xs:element name="GuidHashCount"
      type="int"
      minOccurs="1"
      maxOccurs="1"
     />
    <xs:element name="GuidHash"
      type="GuidHash"
      maxOccurs="unbounded"
      minOccurs="0"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
ServerPublicKey	**String**
GuidHashCount	**int**
GuidHash	**GuidHash**

2.2.5.4.1.1 GuidHash Complex Type

The **GuidHash** complex type defines the parameters returned from an **AcquireTemplateInformation** operation. A valid response MUST include a GUID parameter as a string that represents the GUID of a server template. The response MUST also include a hash parameter as a string that represents the hash of the server template (SHA1 hash of the body element, base64-encoded).

```
<xs:complexType name="TemplateInformation">
  <xs:sequence>
    <xs:element name="Guid"
      type="string"
      maxOccurs="1"
      minOccurs="0"
     />
    <xs:element name="Hash"
      type="string"
      minOccurs="0"
      maxOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
Guid	**string**
Hash	**string**

2.2.5.5 AcquireTemplates Element

The **AcquireTemplates** element contains the body of the request for the **AcquireTemplates** Web method. It MUST include a parameter named guids. This parameter guids is an **string (ArrayOfString)** that represents a list of server template GUIDs. The request indicates the templates that the requestor is interested in obtaining from the server.

```
<xs:element name="AcquireTemplates">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="guids"
        type="string (ArrayOfString)"
        maxOccurs="1"
        minOccurs="0"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
guids	**string (ArrayOfString)**

2.2.5.6 AcquireTemplatesResponse Element

The **AcquireTemplatesResponse Element** contains the response to an AcquireTemplates Web method.

```
<xs:element name="AcquireTemplatesResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AcquireTemplatesResult"
        type="ArrayOfGuideTemplate"
        maxOccurs="1"
        minOccurs="0"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
AcquireTemplatesResult	ArrayOfGuideTemplate

2.2.5.6.1 ArrayOfGuidTemplate Complex Type

The **ArrayOfGuidTemplate** complex type contains any number of elements.

The **ArrayOfGuidTemplate** complex type defines the parameters returned from an **AcquireTemplates** operation. A valid response MUST include **GuidTemplate** parameters of type **GuidTemplate**, each representing a server template. The number of **GuidTemplate** parameters MAY range from 0 to 25.

```
<xs:complexType name="ArrayOfGuidTemplate">
  <xs:sequence>
    <xs:element name="GuidTemplate"
      type="GuidTemplate"
      maxOccurs="unbounded"
      minOccurs="0"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
GuidTemplate	GuidTemplate

2.2.5.6.1.1 GuidTemplate Complex Type

The **GuidTemplate** complex type defines the parameters returned from an **AcquireTemplates** operation. A valid response MUST include a parameter named GUID. The GUID parameter is a string that represents the GUID of a server template. The response MUST also include a hash parameter. The hash parameter is a string that represents the hash of the server template (SHA1 hash of the BODY element, base64-encoded). The response MUST include a template parameter. The template parameter is a string that represents the actual template in serialized XML form.

```
<xs:complexType name="GuidTemplate">
  <xs:sequence>
    <xs:element name="Guid"
      type="string"
      maxOccurs="1"
      minOccurs="0"
     />
    <xs:element name="Hash"
      type="string"
      maxOccurs="1"
      minOccurs="0"
     />
    <xs:element name="Template"
      type="string"
      maxOccurs="1"
      minOccurs="0"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
Guid	**string**
Hash	**string**
Template	**string**

2.2.6 Publishing Service Schema

The complex types, simple types, and elements described in this section are used in the Publishing Service.

2.2.6.1 AcquireIssuanceLicense Element

The AcquireIssuanceLicense element contains the body of the request to the AcquireIssuanceLicense Web method.

```
<xs:element name="AcquireIssuanceLicense">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="RequestParams"
        type="ArrayOfAcquireIssuanceLicenseParams"
```

```
          maxOccurs="1"
          minOccurs="0"
         />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Child Elements

Element	Type
RequestParams	ArrayOfAcquireIssuanceLicenseParams

2.2.6.1.1 ArrayOfAcquireIssuanceLicenseParams Complex Type

The **ArrayOfAcquireIssuanceLicenseParams** complex type defines an array used to provide multiple unsigned issuance licenses as in-parameters to the **AcquireIssuanceLicense** operation.

```
<xs:complexType name="ArrayOfAcquireIssuanceLicenseParams">
  <xs:sequence>
    <xs:element name="AcquireIssuanceLicenseParams"
      type="AcquireIssuanceLicenseParams"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
AcquireIssuanceLicenseParams	**AcquireIssuanceLicenseParams**

2.2.6.1.1.1 AcquireIssuanceLicenseParams Complex Type

The AcquireIssuanceLicenseParams complex type defines the in-parameters for the **AcquireIssuanceLicense** request operation. The in-parameter **UnsignedIssuanceLicense** contains the unsigned issuance license. The license format MUST correspond to the format defined in 2.3.

```
<xs:complexType name="AcquireIssuanceLicenseParams">
  <xs:sequence>
    <xs:element name="UnsignedIssuanceLicense"
      minOccurs="0"
      maxOccurs="1"
    >
      <xs:complexType
        mixed="true"
      >
```

```
        <xs:sequence>
          <xs:any
            namespace=""
           />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
UnsignedIssuanceLicense	N/A

2.2.6.1.1.1.1 UnsignedIssuanceLicense Element

The **UnsignedIssuanceLicense** element contains the issuance license that the client requests the server to sign and is represented as an XmlNode. This license MUST conform to the parameters specified in section 2.3.

```
<xs:element name="UnsignedIssuanceLicense">
  <xs:complexType
    mixed="true"
  >
    <xs:sequence>
      <xs:any
        namespace=""
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.2.6.2 AcquireIssuanceLicenseResponse Element

The **AcquireIssuanceLicenseResponse** element contains the response parameters returned from an **AcquireIssuanceLicense** Web method.

```
<xs:element name="AcquireIssuanceLicenseResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AcquireIssuanceLicenseResult"
        type="ArrayOfAcquireIssuanceLicenseResponse"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
AcquireIssuanceLicenseResult	**ArrayOfAcquireIssuanceLicenseResponse**

2.2.6.2.1 ArrayOfAcquireIssuanceLicenseResponse Complex Type

The **ArrayOfAcquireIssuanceLicenseResponse** complex type contains an array of certificate chains that each represent a signed issuance license.

```
<xs:complexType name="ArrayOfAcquireIssuanceLicenseResponse">
  <xs:sequence>
    <xs:element name="AcquireIssuanceLicenseResponse"
      type="AcquireIssuanceLicenseResponse"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
AcquireIssuanceLicenseResponse	**AcquireIssuanceLicenseResponse**

2.2.6.2.1.1 AcquireIssuanceLicenseResponse Complex Type

The **AcquireIssuanceLicenseResponse** complex type contains an **ArrayOfXmlNode** that contains the signed issuance license issued by the server. The issuance licenses used in this array MUST conform to the format specified in 2.3.

```
<xs:complexType name="AcquireIssuanceLicenseResponse">
  <xs:sequence>
    <xs:element name="CertificateChain"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
CertificateChain	**ArrayOfXmlNode**

2.2.6.3 GetClientLicensorCert Element

The **GetClientLicensorCert** element contains the body of the request used in the **GetClientLicensorCert** Web method request. The **GetClientLicensorCert** operation takes as input one parameter that is an array of user identity certificates.

```
<xs:element name="GetClientLicensorCert">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="RequestParams"
        type="ArrayOfGetClientLicensorCertParams"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
RequestParams	**ArrayOfGetClientLicensorCertParams**

2.2.6.3.1 ArrayOfGetClientLicensorCertParams Complex Type

The **ArrayOfGetClientLicensorCertParams** complex type is an array of **GetClientLicensorCertParams**, each of which contains a set of user identity certificates used in responding to the **GetClientLicensorCert** Web request.

```
<xs:complexType name="ArrayOfGetClientLicensorCertParams">
  <xs:sequence>
    <xs:element name="GetClientLicensorCertParams"
      type="GetClientLicensorCertParams"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
GetClientLicensorCertParams	**GetClientLicensorCertParams**

2.2.6.3.1.1 GetClientLicensorCertParams Complex Type

The **GetClientLicensorCertParams** complex type contains an element named PersonaCerts that is an **ArrayOfXmlNode**, and represents a user identity certificate chain. The GetClientLicensorCert Web method issues CLC chains to the user identities presented via this parameter.

```
<xs:complexType name="GetClientLicensorCertParams">
  <xs:sequence>
    <xs:element name="PersonaCerts"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
PersonaCerts	ArrayOfXmlNode

2.2.6.4 GetClientLicensorCertResponse Element

The **GetClientLicensorCertResponse** element contains the response parameters returned from the GetClientLicensorCertResponse Web method request.

```
<xs:element name="GetClientLicensorCertResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="GetClientLicensorCertResult"
        type="ArrayOfGetClientLicensorCertResponse"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
GetClientLicensorCertResult	ArrayOfGetClientLicensorCertResponse

2.2.6.4.1 ArrayOfGetClientLicensorCertResponse Complex Type

The **ArrayOfGetClientLicensorCertResponse** complex type contains an array of GetClientLicensorCertResponse types that each contain a certificate chain representing a CLC. The CLC grants permissions to the client on behalf of the server so the client can sign issuance licenses itself.

```
<xs:complexType name="ArrayOfGetClIentLicensorCertResponse">
  <xs:sequence>
    <xs:element name="GetClientLicensorCertResponse"
      type="GetClientLicensorCertResponse"
```

```
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
GetClientLicensorCertResponse	**GetClientLicensorCertResponse**

2.2.6.4.1.1 GetClientLicensorCertResponse Complex Type

The **GetClientLicensorCertResponse** complex type contains an **ArrayOfXmlNode** that represents the CLC response from the GetClientLicensorCert Web method request. This CLC MUST conform to the parameters found in 2.3.

```
<xs:complexType name="GetClientLicensorCertResponse">
  <xs:sequence>
    <xs:element name="CertificateChain"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
CertificateChain	**ArrayOfXmlNode**

2.2.7 Server Service Schema

The complex types, simple types, and elements described in this section are used in the Server Service.

2.2.7.1 FindServiceLocationsForUser Element

The **FindServiceLocationsForUser** element contains the body of the message for and is used as an in-parameter to the FindServerLocationsForUser Web method. This element MUST be populated by the client when sending a FindServiceLocations request. The FindServiceLocationsForUser Web method parameters consist of any number of ServiceNames.

```
<xs:element name="FindServiceLocationsForUser">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ServiceNames"
```

```
          type="ArrayOfServiceLocationRequest"
          minOccurs="0"
          maxOccurs="1"
         />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Child Elements

Element	Type
ServiceNames	**ArrayOfServiceLocationRequest**

2.2.7.1.1 ArrayOfServiceLocationRequest Complex Type

The **ArrayOfServiceLocationRequest** complex type is an array of **ServiceLocationRequest** elements.

```
<xs:complexType name="ArrayOfServiceLocationRequest">
  <xs:sequence>
    <xs:element name="ServiceLocationRequest"
      type="ServiceLocationRequest"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
ServiceLocationRequest	**ServiceLocationRequest**

2.2.7.1.1.1 ServiceLocationRequest Complex Type

The **ServiceLocationRequest** complex type contains an enumeration of a service type that indicates a service to locate. Possible values for the enumeration are defined in **ServiceType Simple Type (section 2.2.7.1.1.1.1)**. The enumeration MUST contain a literal string, as specified in **ServiceType Simple Type** (section 2.2.7.1.1.1.1).

```
<xs:complexType name="ServiceLocationRequest">
  <xs:sequence>
    <xs:element name="Type"
      type="ServiceType"
      minOccurs="1"
      maxOccurs="1"
     />
  </xs:sequence>
```

```
</xs:complexType>
```

Child Elements

Element	Type
Type	**ServiceType**

2.2.7.1.1.1.1 ServiceType Simple Type

The **ServiceType** simple type enumerates each of the possible types of service that a rights management server may provide. The **ServiceType** simple type is used to enumerate the type of service or services offered by a rights management server and is part of both the in- and out-parameters of the **FindServiceLocationsForUser** operation.

Version 1 of the RMS: Client-to-Server Protocol introduced the **FindServiceLocationsForUser** and the **ServiceType** simple type consisting of enumeration values for: EnrollmentService, LicensingService, PublishingService, CertificationService, ActivationService, PrecertificationService, ServiceService and DrmRemoteDirectoryServices.

Windows XP SP2and versions of the client introduced the following new enumeration values: GroupExpansionService, LicensingInternalService, and CertificationInternalService.

```
<xs:simpleType name="ServiceType">
  <xs:restriction
    base="string"
  >
    <xs:enumeration
      value="EnrollmentService"
     />
    <xs:enumeration
      value="LicensingService"
     />
    <xs:enumeration
      value="PublishingService"
     />
    <xs:enumeration
      value="CertificationService"
     />
    <xs:enumeration
      value="ActivationService"
     />
    <xs:enumeration
      value="PrecertificationService"
     />
    <xs:enumeration
      value="ServerService"
     />
    <xs:enumeration
      value="DrmRemoteDirectoryServices"
     />
    <xs:enumeration
      value="GroupExpansionService"
     />
```

```
      <xs:enumeration
        value="LicensingInternalService"
       />
      <xs:enumeration
        value="CertificationInternalService"
       />
    </xs:restriction>
  </xs:simpleType>
```

Enumeration

The following values are defined by the **ServiceType** simple type:

Value	Description
EnrollmentService	Enumerates the Enrollment service.
LicensingService	Enumerates the Licensing service.
PublishingService	Enumerates the Publishing service.
CertificationService	Enumerates the Certification service.
ActivationService	Enumerates the Activation service.
PrecertificationService	Enumerates the PreCertification service.
ServerService	Enumerates the Server service.
DrmRemoteDirectoryServices	Enumerates the DrmRemoteDirectory service.
GroupExpansionService	Enumerates the Group Expansion Service. Present in Windows XP SP2 and Windows Vista.
LicensingInternalService	Enumerates the internal Licensing service. Enumerates the internal Licensing Service. Present in Windows XP SP2 and Windows Vista clients.
CertificationInternalService	Enumerates the internal Certification Service. Present in Windows XP SP2 and Windows Vista clients.

2.2.7.2 FindServiceLocationsForUserResponse Element

The **FindServiceLocationsForUserResponse** element is a complex type that contains an array of service location response elements. This element is used as an out-parameter for the **FindServiceLocationsForUserResponse** operation.

```
<xs:element name="FindServiceLocationsForUserResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="FindServiceLocationsForUserResult"
        type="ArrayOfServiceLocationResponse"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
```

```
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
FindServiceLocationsForUserResult	**ArrayOfServiceLocationResponse**

2.2.7.2.1 ArrayOfServiceLocationResponse Complex Type

The **ArrayOfServiceLocationResponse** complex type contains an array of **ServiceLocationResponse** types. This array is used to respond to a **FindServiceLocationsForUser** operation.

```
<xs:complexType name="ArrayOfServiceLocationResponse">
  <xs:sequence>
    <xs:element name="ServiceLocationResponse"
      type="ServiceLocationResponse"
      minOccurs="0"
      maxOccurs="unbounded"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
ServiceLocationResponse	**ServiceLocationResponse**

2.2.7.2.1.1 ServiceLocationResponse Complex Type

The **ServiceLocationResponse** complex type contains a URL that is associated with an RMS and the type of that service. The URL MUST be a literal string. The Type element MUST be a literal string from the set of possible values for **ServiceType**.

```
<xs:complexType name="ServiceLocationResponse">
  <xs:sequence>
    <xs:element name="URL"
      type="string"
      minOccurs="0"
      maxOccurs="1"
     />
    <xs:element name="Type"
      type="ServiceType"
      minOccurs="1"
      maxOccurs="1"
     />
  </xs:sequence>
```

```
</xs:complexType>
```

Child Elements

Element	Type
URL	**string**
Type	**ServiceType**

2.2.7.3 GetLicensorCertificate Element

The **GetLicensorCertificate** element contains the body of the request for the GetLicensorCertificate Web method. This element MUST NOT contain any elements.

```
<xs:element name="GetLicensorCertificate">
  <xs:complexType />
</xs:element>
```

2.2.7.4 GetLicensorCertificateResponse Element

The **GetLicensorCertificateResponse** element is a complex data type that contains the response data returned from a GetLicensorCertificate operation. The certificate chain included here MUST correspond to the certificate formats found in 2.3.

```
<xs:element name="GetLicensorCertificateResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="GetLicensorCertificateResult"
        type="LicensorCertChain"
        minOccurs="0"
        maxOccurs="1"
       />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Child Elements

Element	Type
GetLicensorCertificateResult	**LicensorCertChain**

2.2.7.4.1 LicensorCertChain Complex Type

The **LicensorCertChain** complex type represents a set of certificates that are related to each other by successive issuers. For example, in a **LicensorCertChain** instance that contains A, B, and C certificates, A is issued by B, and B is issued by C.

```
<xs:complexType name="LicensorCertChain">
  <xs:sequence>
    <xs:element name="CertificateChain"
      type="ArrayOfXmlNode"
      minOccurs="0"
      maxOccurs="1"
     />
  </xs:sequence>
</xs:complexType>
```

Child Elements

Element	Type
CertificateChain	ArrayOfXmlNode

2.3 Certificate Formats

This section describes the way the RMS: Client-to-Server Protocol utilizes [XrML] for certificates and licenses.

2.3.1 Common Certificate and License Structures

This section describes in detail common sections of RMS certificate formats. All elements MUST follow the [XrML] schema.

2.3.1.1 ISSUEDTIME

The ISSUEDTIME element specifies the time that a certificate or license was generated, expressed in **UTC**. ISSUEDTIME is specified in the XrML DTD. All certificates and licenses MUST contain an ISSUEDTIME element.

An ISSUEDTIME element MUST follow this template.

```
<ISSUEDTIME>
   [[- issuedtime -]]
</ISSUEDTIME>
```

[[- issuedtime -]]: The time at which the certificate or license was generated, expressed in UTC.

2.3.1.2 VALIDITYTIME

VALIDITYTIME is an optional element that specifies the time period in which a certificate or license can be used. The certificate or license MUST be considered invalid outside this time period. The time period is a half-closed interval in which the start time is included in the set but the end time is not. A certificate or license MAY contain a VALIDITYTIME element.

A VALIDITYTIME element MUST use the following template.

```
<VALIDITYTIME>
   <FROM>[[- starttime -]]</FROM>
```

```
    <UNTIL>[[- endtime -]]</UNTIL>
</VALIDITYTIME>
```

[[- starttime -]]: The beginning of the time interval in which the certificate is allowed to be considered valid, expressed in UTC.

[[- endtime -]]: The end of the time interval in which the certificate is allowed be considered valid, expressed in UTC.

2.3.1.3 RANGETIME

RANGETIME specifies a time condition on the ability to exercise a right that is granted in a certificate or license. The time period is a half-closed interval in which the start time is included in the set but the end time is not.

The RANGETIME element MUST use the following template.

```
<RANGETIME>
  <FROM>[[- starttime -]]</FROM>
  <UNTIL>[[- endtime -]]</UNTIL>
</RANGETIME>
```

[[- starttime -]]: The beginning of the time period in which a right is allowed to be exercised, expressed in UTC.

[[- endtime -]]: The end of the time period in which a right is allowed to be exercised, expressed in UTC.

2.3.1.4 DESCRIPTOR

The DESCRIPTOR element identifies the certificate or license and describes its type. All certificates and licenses MUST contain a DESCRIPTOR element.

The DESCRIPTOR element MUST use the following template.

```
<DESCRIPTOR>
  [[- object -]]
</DESCRIPTOR>
```

[[- object -]]: An object that identifies the certificate or license. An object is specified in the XrML DTD. Specific content is defined for each certificate and license

2.3.1.5 ISSUER

The ISSUER element describes the entity that issued or signed the certificate or license. All certificates and licenses MUST contain an <ISSUER> element. The ISSUER element MUST contain an object element that identifies the issuer along with a PUBLICKEY (section 2.3.1.6) element that contains the issuer's public key.

An ISSUER element MUST use the following template.

```
<ISSUER>
  [[- object -]]
  [[- publickey -]]
  [[- optionalinfo -]]
</ISSUER>
```

[[- object -]]: An object that identifies the issuer. An object is specified in the XrML DTD. Specific content of the object depends on the certificate or license.

[[- publickey -]]: The issuer's public key contained in a PUBLICKEY element.

[[- optionalinfo -]]: Optional information about the issuer. Specific content is defined for each certificate and license.

2.3.1.6 PUBLICKEY

A PUBLICKEY element contains an RSA public key. A PUBLICKEY element MUST use the following template.

```
<PUBLICKEY>
   <ALGORITHM>RSA</ALGORITHM>
   <PARAMETER name="public-exponent">
      <VALUE encoding="integer32">
         [[- exponent -]]
      </VALUE>
   </PARAMETER>
   <PARAMETER name="modulus">
      <VALUE encoding="base64" size="[[- key length -]]">
         [[- modulus -]]
      </VALUE>
   </PARAMETER>
</PUBLICKEY>
```

[[- exponent -]]: The exponent portion of the public key. This MUST be a valid exponent for the RSA algorithm, and always be set to 65537.

[[- key length -]]: The length of the public key in bits, represented as a string. This MUST be a valid key length for the RSA algorithm.

[[- modulus -]]: The modulus portion of the public key. This MUST be a valid modulus for the RSA algorithm.

2.3.1.7 DISTRIBUTIONPOINT

A DISTRIBUTIONPOINT element is optional and describes an address or location for a particular service. A certificate or license MAY contain multiple DISTRIBUTIONPOINT elements.

A DISTRIBUTIONPOINT element MUST use the following template.

```
<DISTRIBUTIONPOINT>
   [[- object -]]
   [[- publickey -]]
```

```
</DISTRIBUTIONPOINT>
```

[[- object -]]: An object that identifies the DISTRIBUTIONPOINT. An object is specified in the XrML DTD. Specific content is defined for each certificate and license.

[[- publickey -]]: MAY be present if the object element of the DISTRIBUTIONPOINT element is of type "Revocation". MUST NOT be present otherwise. If present, this MUST contain one PUBLICKEY (section 2.3.1.6) element.

2.3.1.8 NAME

A NAME element contains a friendly name.

A NAME element MUST use the following template.

```
<NAME>
   [[- name -]]
</NAME>
```

[[- name -]]: A string. The value of this placeholder depends on the specific application in a certificate or license and is defined explicitly for each certificate and license format.

2.3.1.9 ADDRESS

An ADDRESS element contains a URL address.

An ADDRESS element MUST use the following template.

```
<ADDRESS type="[[- type -]]">
   [[- address -]]
</ADDRESS>
```

[[- type -]]: A string containing a type of address that can take the value of "URL" or "email_alias". The value of this placeholder depends on the specific application in a certificate or license and is defined explicitly for each certificate and license format.

[[- address -]]: A string containing the address. The value of this placeholder depends on the specific application in a certificate or license and is defined explicitly for each certificate and license format.

2.3.1.10 SECURITYLEVEL

A SECURITYLEVEL element contains additional information in a name/value pair. A SECURITYLEVEL element MUST follow the XrML DTD.

A SECURITYLEVEL element MUST use the following template.

```
<SECURITYLEVEL name="[[- name -]]" value="[[- value -]]"/>
```

[[- name -]]: An arbitrary string containing the name of the name/value pair. The value of this placeholder depends on the specific application in a certificate or license and is defined explicitly for each certificate and license format.

[[- value -]]: An arbitrary string containing the value of the name/value pair. The value of this placeholder depends on the specific application in a certificate or license and is defined explicitly for each certificate and license format.

2.3.1.11 ISSUEDPRINCIPALS

For a certificate, the ISSUEDPRINCIPALS element describes the role, identity, and key being issued by the certificate. For a license, the ISSUEDPRINCIPALS element describes the principal to which rights are being granted. All certificates and licenses MUST contain an ISSUEDPRINCIPALS element. An ISSUEDPRINCIPALS element MUST contain exactly one principal.

An ISSUEDPRINCIPALS element MUST use the following template.

```
<ISSUEDPRINCIPALS>
   <PRINCIPAL internal-id="1">
      [[- object -]]
      [[- publickey -]]
      [[- digest -]]
      [[- optionalinfo -]]
      [[- enablingbits -]]
   </PRINCIPAL>
</ISSUEDPRINCIPALS>
```

[[- object -]]: An object that identifies the principal. An object is specified in the XrML DTD. The value of this placeholder depends on the specific application in a certificate or license and is defined explicitly for each certificate and license format.

[[- publickey -]]: The public key of a principal contained in a PUBLICKEY element. For certificates, this is the public key being issued to the principal. For licenses, this is an existing public key that has already been issued to the principal.

[[-digest -]]: An SPC MUST include a digest element containing a hardware ID hash. All other certificates and licenses MUST NOT include a digest element here.

[[- optionalinfo -]]: Other information MAY be included in the form of SECURITYLEVEL elements.

[[- enablingbits -]]: A publishing license MUST include an ENABLINGBITS element that contains the encrypted rights data. All other certificates and licenses MUST NOT include an ENABLINGBITS element here.

2.3.1.12 SIGNATURE

The SIGNATURE element contains the cryptographic signature of a license or certificate and is appended to the end of each license or certificate. It is computed from the body element of the license or certificate that it is contained in, including the body tags, and follows the format specified by XrML.

A SIGNATURE element MUST use the following template.

```
<SIGNATURE>
```

```
    <ALGORITHM>RSA PKCS#1-V1.5</ALGORITHM>
    <DIGEST>
       <ALGORITHM>SHA1</ALGORITHM>
       <PARAMETER name="codingtype">
          <VALUE encoding="string">
             surface-coding
          </VALUE>
       </PARAMETER>
       <VALUE encoding="base64" size="160">
          [[- hash -]]
       </VALUE>
    </DIGEST>
    <VALUE encoding="base64" size="[[- size -]]">
       [[- signature -]]
    </VALUE>
 </SIGNATURE>
```

[[- hash -]]: The SHA1 hash of the body element, base64-encoded.

[[- size -]]: The size, in bits, of the Issuer's private key that was used to compute the signature, represented as a string.

[[- signature -]]: The SHA1 hash of the body element, encrypted with the issuer's private key, base64-encoded.

2.3.1.13 ENABLINGBITS

An ENABLINGBITS element includes a key and a hash encrypted together in a license or certificate. The format for ENABLINGBITS is as follows:

1. Enabling bits in XrML license = Base64Encoded(RawEnablingBits)
2. RawEnablingBits = K_{Public}(KeyHeader & $K_{Session}$) + $K_{Session}$(EnablingBitsHeader + (KeyHeader & K) + Hash)

Note Notation: 'K(A)' means data 'A' encrypted with key 'K'.

License	KPublic	K	160 bit SHA-1 Hashed Data
PL	Licensor (RMS Server) public key (1,024-bit RSA)	Symmetric content key (128-bit **AES**)	ISSUEDPRINCIPALS element of PL
UL	RAC public key (1,024-bit RSA)	Symmetric content key (128-bit AES)	ISSUER element of UL
CLC Chain	RAC public key (1,024-bit RSA)	CLC private key (1,024-bit RSA)	ISSUER element of CLC
RAC	Security processor public key (1,024-bit RSA)	RAC private key (1,024-bit RSA)	ISSUER element of RAC

K varies depending upon the type of license. The preceding table describes what K and A are for each of the license types that contain enabling bits.

The session key can be either a 56-bit **Data Encryption Standard (DES)** key or a 128-, 192-, or 256-bit Advanced Encryption Standard (AES) key. The KeyHeader for the session key describes the key type, size, and block size. For more information about the KeyHeader, see section 2.3.1.13.1.

A new session key is randomly generated each time the client or server has to create enabling bits. The session key is encrypted with the public key (licensor public key, group identity certificate [GIC] public key, or machine public key, depending upon the license type) and this forms the first 1,024 bits of the ENABLINGBITS, assuming a 1,024-bit RSA key was used for the encryption. The size of this equals the size of the RSA key pair encrypting the symmetric key, and since during decryption the size of the private key is already known (from the prologue of the key bits), the size of the encrypted symmetric key is also known.

The session key is used to encrypt the rest of the data in the ENABLINGBITS. The rest of the data includes an enabling bits header, the key header and key, and the hash.

The ENABLINGBITS header is defined as follows:

```
typedef struct _UDEBHeader
{
    DWORD dwVersion;
    DWORD dwcbSize;
    DWORD dwReserved1;
    DWORD dwReserved2;
} UDEBHeader;
```

The dwVersion most currently used is defined as follows:

```
#define UDEBHEADER_VERSION_CURRENT 0x1
```

The size of the header is 128 bits. The dwcbSize indicates the combined size of the payload and hash.

The key itself will either be a 1,024-bit RSA private key or a 56-bit DES or AES (128-, 192-, or 256-bit) content symmetric key. The KeyHeader in front of the key specifies the key type, size, and algorithm block size.

The hash is a 160-bit SHA1 hash of XrML data. The XrML data that is hashed depends upon the type of XrML document, as described in the preceding table.

The ENABLINGBITS header, the payload, and the hash are concatenated and then encrypted with the freshly generated symmetric key. The result of this encryption is then concatenated with the encrypted symmetric key, and the result of this is base64-encoded and can be inserted into the XrML document.

The ENABLINGBITS element contains the enabling bits in XrML. It MUST follow the XrML DTD and the following template:

```
<ENABLINGBITS type="sealed-key">
   <VALUE encoding="base64" size="[[- size -]]">
      [[- sealedkey -]]
   </VALUE>
</ENABLINGBITS>
```

[[- size -]]: The length of the enabling bits.

[[- sealedkey -]]: The enabling bits, base64-encoded.

2.3.1.13.1 KeyHeader

The KeyHeader for the session key describes the key type, size, and block size for the algorithm as detailed in the following table.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BlobSize																Reserved															
keySizeInBytes																blockSizeInBytes															
Flags																															
Key																															

BlobSize (2 bytes): The BlobSize field MUST be the size, in bytes, of the complete KeyHeader plus <Key> structure.

Reserved (2 bytes): The Reserved field MUST be set to 0xFFFF.

keySizeInBytes (2 bytes): The keySizeInBytes field MUST be the symmetric key size in bits. For DES, this MUST be 56. For AES (Rijndael) size MUST be either 128 (the default), 192, or 256 bits.

blockSizeInBytes (2 bytes): The BlockSizeInBytes field is the key block size, which varies depending on the cryptographic provider.

Flags (4 bytes): The Flags field is a bit field with the following structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C	0	0	0	0	0	0	0	0	0	0	0	A

Where the bits are defined as:

Value	Description
C Cipher Mode	The Cipher Mode bit MUST be set to 1 to indicate Electronic Codebook.
A Algorithm	The Algorithm bit MUST be set to 0 if the key is a DES key. The Algorithm bit MUST be set to 1 if the key is an AES key.

Key (4 bytes): This field contains a key as per the flags in the preceding fields.

2.3.2 Certificate and License Chains

A certificate or license chain shows the issuing and trust hierarchy for a given certificate or license. The following diagram explains the relationships between certificates.

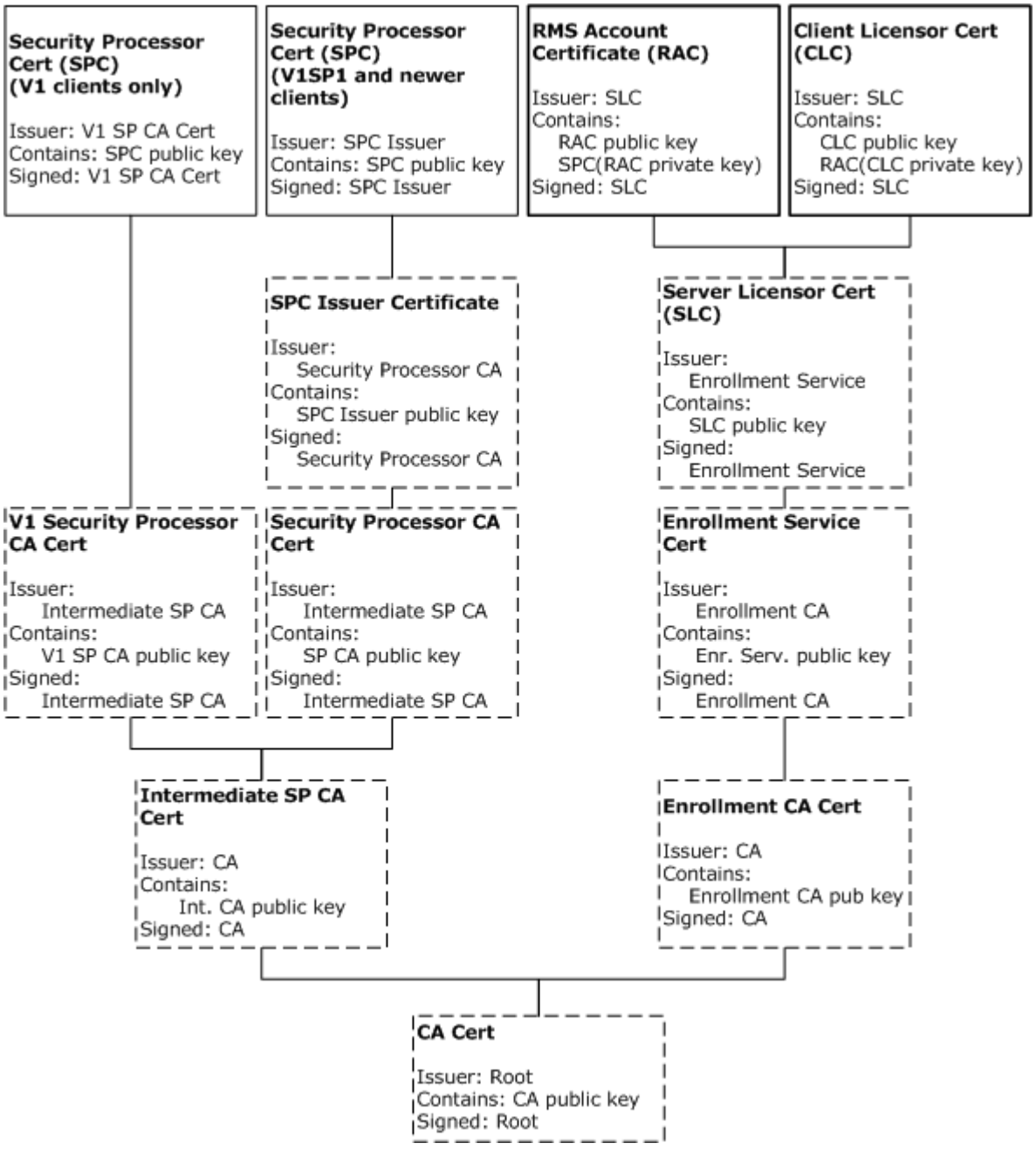

Figure 4: Relationships between certificates

For version 1 clients, the SPC chain starts at the SPC leaf node certificate, followed by the version 1 security processor **Certification Authority (CA)** certificate, followed by the intermediate security processor CA certificate, and terminates at the CA certificate. For version 1 SP1 and newer clients, the SPC chain starts at the SPC leaf node certificate, followed by the SPC Issuer certificate, followed by the security processor CA certificate, followed by the intermediate security processor CA certificate, and terminates at the CA certificate. Certificates in the SPC chain are acquired during client machine activation and are never generated by the server. For more information on client machine activation, see 3.2.3.1.

The RAC chain starts at the RAC leaf node certificate, followed by the SLC, followed by the Enrollment Service certificate, followed by the Enrollment CA certificate, terminating at the CA certificate. The CLC chain starts at the CLC leaf node certificate, followed by the SLC, followed by the Enrollment Service certificate, followed by the Enrollment CA certificate, and terminating at the CA certificate.

Certificates in dark boxes (RAC and CLC) are issued by the server. Certificates from the SLC and below are acquired during server enrollment. For more information on server enrollment, see 3.1.3.2.1.

Certificates in dashed boxes (SLC, version 1 security processor CA certificate, SPC Issuer certificate, security processor CA certificate, intermediate security processor CA certificate, CA certificate, Enrollment Service certificate, and Enrollment CA certificate) are issuing certificates and follow a similar format.

The following diagram explains the relationships between licenses and the certificate in their chains.

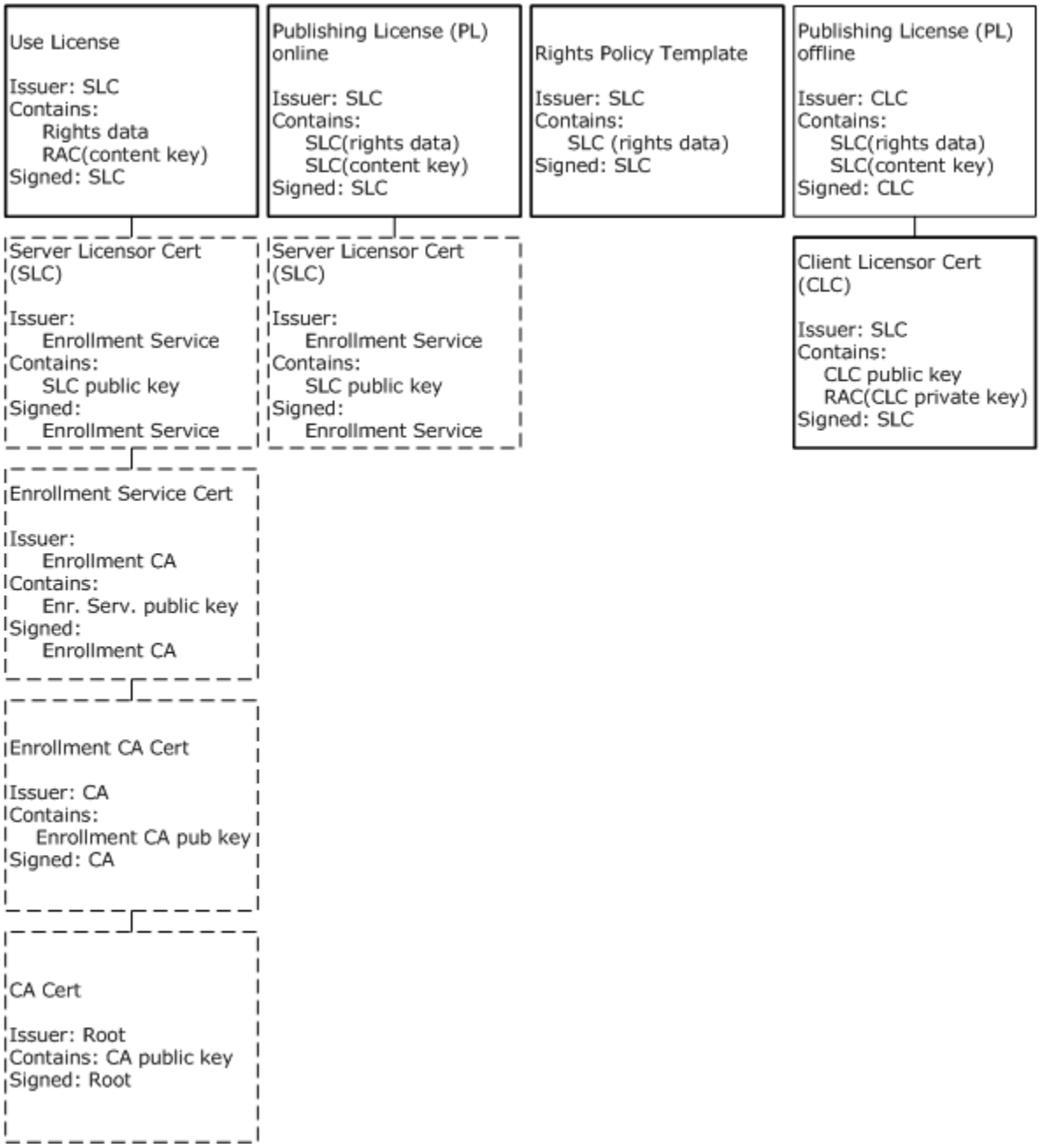

Figure 5: Relationships between licenses and certificates

The UL chain starts at the UL leaf node certificate, followed by the SLC, followed by the Enrollment Service certificate, followed by the Enrollment CA certificate, terminating at the CA certificate.

For content published online, the PL chain starts at the PL leaf node certificate and terminates at the SLC. For content published offline, the PL chain starts at the PL leaf node certificate and terminates at the CLC.

The rights policy template is signed by the SLC, but exists as a single-node certificate.

Licenses in dark boxes (UL and online PL) are issued by the server. The offline PL is issued by the client.

2.3.3 Issuing Certificates

This section defines the format of issuing certificates. The SLC, version 1 security processor CA certificate, SPC issuer certificate, security processor CA certificate, intermediate security processor CA certificate, CA certificate, Enrollment Service certificate, and Enrollment CA certificate, are all Issuing certificates.

Issuing certificates MUST use the following template.

```
<XrML xmlns="" version="1.2">
   <BODY type="LICENSE" version="3.0">
      [[- issuedtime -]]
      [[- validitytime -]]
      [[- descriptor -]]
      [[- issuer -]]
      [[- issuedprincipals -]]
      <WORK>
         [[- workobject -]]
         <RIGHTSGROUP name="Main-Rights">
            <RIGHTSLIST>
               <RIGHT name="ISSUE">
                  <CONDITIONLIST>
                     <TIME>
                        [[- rangetime -]]
                     </TIME>
                     <ACCESS>
                        <PRINCIPAL internal-id="1" />
                     </ACCESS>
                  </CONDITIONLIST>
               </RIGHT>
            </RIGHTSLIST>
         </RIGHTSGROUP>
      </WORK>
      [[- conditionlist -]]
   </BODY>
   [[- signature -]]
</XrML>
```

[[- issuedtime -]]: MUST be an ISSUEDTIME (section 2.3.1.1) element containing the time the certificate was generated, in UTC. The time MUST fall within the RANGETIME of the issuer's certificate.

[[- validitytime -]]: SHOULD be a VALIDITYTIME (section 2.3.1.2) element describing the period of validity for the certificate, in UTC. This element MAY be present but is optional.

[[- descriptor -]]: MUST be a DESCRIPTOR (section 2.3.1.4) element describing the certificate.

[[- issuer -]]: MUST be an ISSUER (section 2.3.1.5) element describing the issuer of the certificate.

[[- issuedprincipals -]]: MUST be an ISSUEDPRINCIPALS (section 2.3.1.11) element describing the principal and its public key.

[[- workobject -]]: MUST be an OBJECT element that identifies the certificate. Copied verbatim from the OBJECT in the DESCRIPTOR (section 2.3.3.1) including the same GUID. This OBJECT is described in the DESCRIPTOR (section 2.3.3.1) section.

[[- rangetime -]]: MUST be a RANGETIME (section 2.3.1.3) element describing the period during which the certificate can be used for issuance.

[[- conditionlist -]]: MAY be present in the SLC only. MUST NOT be present in other issuing certificates. If present, this MUST be a CONDITIONLIST (section 2.3.3.4) element that specifies alternate revocation information.

[[- signature -]]: MUST be a SIGNATURE (section 2.3.1.12) element containing the cryptographic signature of the body of the certificate, generated by the issuer of the certificate. The hash MUST be the SHA1 hash of the body. The signature MUST be the hash encrypted with the issuer's private key. The key length MUST be the length of the issuer's private key, which MUST match the length of the issuer's public key.

2.3.3.1 DESCRIPTOR

The DESCRIPTOR element of Issuing certificates describes the type of the certificate and MUST use the following template.

```
<DESCRIPTOR>
   <OBJECT type="[[- type -]]">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
   </OBJECT>
</DESCRIPTOR>
```

[[- type -]]: MUST contain the literal string from the following table.

Certificate	Literal String
SLC	Server-Licensor-Certificate
Enrollment Service Certificate	Server-Licensor-Certificate
Enrollment CA certificate	DRM-CA-Certificate
Version 1 security processor CA certificate	Server-Licensor-Certificate
SPC issuer certificate	Server-Licensor-Certificate
Security processor CA certificate	DRM-CA-Certificate
Intermediate Security Processor CA Certificate	DRM-CA-Certificate
CA certificate	DRM-CA-Certificate

[[- GUID -]]: MUST be a unique GUID that identifies the certificate, represented as a literal **ASCII** string enclosed in braces.

2.3.3.2 ISSUER

The ISSUER element of issuing certificates identifies the issuer of the certificate and MUST use the following template. The contents are generally copied from the principal in the ISSUEDPRINCIPALS element of the issuer's certificates.

```
<ISSUER>
   <OBJECT type="[[- objecttype -]]">
      <ID type="[[- idtype -]]">
         [[- id -]]
      </ID>
      [[- name -]]
   </OBJECT>
   [[- publickey -]]
   [[- cps -]]
</ISSUER>
```

[[- objecttype -]]: MUST contain the literal string found in the following table, specifying the type of the issuer.

Certificate	Literal string
SLC	MS-DRM-Server
Enrollment Service certificate	DRM-Certificate-Authority
Enrollment CA certificate	DRM-Certificate-Authority
Version 1 security processor CA certificate	DRM-Certificate-Authority
SPC issuer certificate	DRM-Desktop-Security-Processor-Certificate-Authority
Security processor CA certificate	DRM-Certificate-Authority
Intermediate security processor CA certificate	DRM-Certificate-Authority
CA certificate	DRM-Certificate-Authority

[[- idtype -]]: MUST contain the literal string found in the following table, specifying the type of identifier used to identify the issuer.

Certificate	Literal string
SLC)	MS-GUID
Enrollment Service certificate	ascii-tag
Enrollment CA certificate	ascii-tag
Version 1 security processor CA certificate	ascii-tag
SPC issuer certificate	MS-GUID
Security processor CA certificate	ascii-tag
Intermediate security processor CA certificate	ascii-tag

Certificate	Literal string
CA certificate	ascii-tag

[[- id -]]: MUST contain the value or literal string from the following table, identifying the issuer. The *[[- GUID -]]* placeholder is defined immediately following the table.

Certificate	Literal string
SLC	*[[- GUID -]]*
Enrollment Service certificate	Microsoft DRM Production Server Enrollment CA
Enrollment CA certificate	Microsoft DRM Production CA
Version 1 security processor CA certificate	Microsoft DRM Production Machine Activation Server CA
SPC issuer certificate	*[[- GUID -]]*
Security processor CA certificate	Microsoft DRM Production Machine Activation Server CA
Intermediate security processor CA certificate	Microsoft DRM Production CA
CA certificate	Microsoft DRM Production Root

[[- GUID -]]: A unique GUID that identifies the issuer of the certificate, represented as a literal ASCII string enclosed in braces. MUST be taken from the object of the principal of the ISSUEDPRINCIPALS of the issuer's certificate.

[[- name -]]: SHOULD be a name element containing the literal string from the following table, specifying a name for the issuer.

Certificate	Literal string
SLC	Microsoft DRM Server Enrollment Service
Enrollment Service certificate	Microsoft DRM Production Server Enrollment CA
Enrollment CA certificate	Microsoft DRM Production CA
Version 1 security processor CA certificate	Microsoft DRM Production Machine Activation Server CA
SPC issuer certificate	Microsoft DRM Production Machine Activation Desktop Security Processor CA
Security processor CA certificate	Microsoft DRM Production Machine Activation Server CA
Intermediate security processor CA certificate	Microsoft DRM Production CA
CA certificate	Microsoft DRM Production Root

[[- publickey -]]: MUST be a PUBLICKEY element that contains the issuer's public key. Exponent MUST be set to 65537. Modulus MUST contain the modulus of the issuer's public key. Size MUST be specified in bits and MUST follow this table.

Certificate	Literal string
SLC	1024
Enrollment Service certificate	1024
Enrollment CA certificate	2048
Version 1 security processor CA certificate	1024
SPC issuer certificate	1024
Security processor CA certificate	1024
Intermediate security processor CA certificate	2048
CA certificate	2048

[[- cps -]]: SHOULD be found in the SLC but MUST NOT be found in any other certificates. The SLC MAY contain a SECURITYLEVEL element with the name "Certificate Practice Statement" and value of a URL pointing to a certificate practice statement.

2.3.3.3 ISSUEDPRINCIPALS

The ISSUEDPRINCIPALS element of an issuing certificate describes the role, identity, and key the certificate is issuing. It MUST use the following template.

```
<ISSUEDPRINCIPALS>
   <PRINCIPAL internal-id="1">
      <OBJECT type="[[- objecttype -]]">
         <ID type="[[- idtype -]]">
            [[- id -]
         </ID>
         [[- name -]]
         [[- address -]]
      </OBJECT>
      [[- publickey -]]
      [[- serverversion -]]
      [[- serversku -]]
   </PRINCIPAL>
</ISSUEDPRINCIPALS>
```

[[- objecttype -]]: MUST contain the literal string, as listed in the following table, specifying the type of principal the certificate is issuing.

Certificate	Literal string
SLC	MS-DRM-Server
Enrollment Service certificate	MS-DRM-Server
Enrollment CA certificate	DRM-Certificate-Authority
Version 1 security processor CA certificate	MS-DRM-Server

Certificate	Literal string
SPC issuer certificate	MS-DRM-Desktop-Security-Processor
Security processor CA certificate	DRM-Desktop-Security-Processor-Certificate-Authority
Intermediate security processor CA certificate	DRM-Certificate-Authority
CA certificate	DRM-Certificate-Authority

[[- idtype -]]: MUST contain the literal string, as listed in the following table, specifying the type of identifier used to identify the principal.

Certificate	Literal string
SLC	MS-GUID
Enrollment Service certificate	MS-GUID
Enrollment CA certificate	ascii-tag
Version 1 security processor CA certificate	MS-GUID
SPC issuer certificate	MS-GUID
Security processor CA certificate	MS-GUID
Intermediate security processor CA certificate	ascii-tag
CA certificate	ascii-tag

[[- id -]]: MUST contain the value or literal string, as listed in the following table, identifying the principal. The *[[- GUID -]]* placeholder is defined immediately following the table.

Certificate	String
SLC	*[[- GUID -]]*
Enrollment Service certificate	*[[- GUID -]]*
Enrollment CA certificate	Microsoft DRM Production Server Enrollment CA
Version 1 security processor CA certificate	*[[- GUID -]]*
SPC issuer certificate	*[[- GUID -]]*
Security processor CA certificate	*[[- GUID -]]*
Intermediate security processor CA certificate	Microsoft DRM Production Machine Activation Server CA
CA certificate	Microsoft DRM Production CA

[[- GUID -]]: MUST be a unique GUID that identifies the principal the certificate is issuing, represented as a literal ASCII string enclosed in braces.

[[- name -]]: MUST be present in all issuing certificates except for the SLC. MUST NOT be present in the SLC. MUST be a name element containing the literal string, as listed in the following table, specifying a name for the principal.

Certificate	String
Enrollment Service certificate	Microsoft DRM Server Enrollment Service
Enrollment CA certificate	Microsoft DRM Production Server Enrollment CA
Version 1 security processor CA certificate	Microsoft DRM Machine Activation Service
SPC issuer certificate	Microsoft DRM Production Desktop Security Processor Activation Certificate
Security processor CA certificate	Microsoft DRM Production Machine Activation Desktop Security Processor CA
Intermediate security processor CA certificate	Microsoft DRM Production Machine Activation Server CA
CA certificate	Microsoft DRM Production CA

[[- address -]]: MUST be present in the SLC only. MUST NOT be present in other issuing certificates. MUST be an address element of type "URL" containing the URL of the server.

[[- publickey -]]: MUST contain the public key being issued. Exponent MUST be set to 65537. Modulus MUST contain the modulus of the public key. Size MUST be specified in bits, as indicated in the following table.

Certificate	String
SLC	1024
Enrollment Service certificate	1024
Enrollment CA certificate	1024
Version 1 security processor CA certificate	1024
SPC issuer certificate	1024
Security processor CA certificate	1024
Intermediate security processor CA certificate	1024
CA certificate	2048

[[- serverversion -]]: SHOULD be present in the SLC only. MUST NOT be present in other issuing certificates. SHOULD be a SECURITYLEVEL element with the name "Server-Version" and the value of a string that contains the version string of the server.

[[- serversku -]]: SHOULD be present in the SLC only. MUST NOT be present in other issuing certificates. SHOULD be a SECURITYLEVEL element with the name "Server-SKU" and the value of a string that contains further version or edition information of the server.

2.3.3.4 CONDITIONLIST

If the SLC was issued with custom revocation authorities specified, it MAY contain a CONDITIONLIST element that describes one or more revocation authorities with its public key.

The CONDITIONLIST element MUST use the following template.

```
<CONDITIONLIST>
   <REFRESH>
      [[- distributionpoint1 -]]
      [[- distributionpoint2 -]]
      <INTERVALTIME />
   </REFRESH>
</CONDITIONLIST>
```

[[- distributionpoint1 -]]: MUST be a DISTRIBUTIONPOINT (section 2.3.3.5) element that contains the public key of the issuer of the SLC, as specified in DISTRIBUTIONPOINT.

[[- distributionpoint2 -]]: MUST contain at least one DISTRIBUTIONPOINT element that contains the public key of a third-party revocation authority that is allowed to revoke the SLC. MAY include additional DISTRIBUTIONPOINT elements as peers, with one element for each revocation authority, as specified in DISTRIBUTIONPOINT.

2.3.3.5 DISTRIBUTIONPOINT

The DISTRIBUTIONPOINT elements in the CONDITIONLIST describe the public keys of revocation authorities who are authorized to revoke the SLC. The DISTRIBUTIONPOINT elements MUST use the following template.

```
<DISTRIBUTIONPOINT>
   <OBJECT type="Revocation">
      <ID type="ascii-tag">
         External revocation authority
      </ID>
   </OBJECT>
   [[- publickey -]]
</DISTRIBUTIONPOINT>
```

[[- publickey -]]: MUST be a PUBLICKEY (section 2.3.1.6) element that contains the public key of the revocation authority.

2.3.4 Security Processor Certificate

This section defines the format of the security processor certificate (SPC). The SPC is acquired during client initialization and is never generated by the server.

The SPC MUST use the following template.

```
<XrML version="1.2" xmlns="">
   <BODY type="LICENSE" version="3.0">
      [[- issuedtime -]]
      [[- descriptor -]]
      [[- issuer -]]
      [[- distributionpoint -]]
      [[- issuedprincipals -]]
   </BODY>
   [[- signature -]]
</XrML>
```

[[- issuedtime -]]: MUST be an ISSUEDTIME (section 2.3.1.1) element containing the time the SPC was generated, in UTC.

[[- descriptor -]]: MUST be a DESCRIPTOR (section 2.3.1.4) element describing the SPC.

[[- issuer -]]: MUST be an ISSUER (section 2.3.1.5) element describing the issuer of the SPC.

[[- distributionpoint -]]: MUST be a DISTRIBUTIONPOINT (section 2.3.1.7) element describing the issuer of the SPC.

[[- issuedprincipals -]]: MUST be an ISSUEDPRINCIPALS (section 2.3.1.11) element describing the principal and the SPC public key.

[[- signature -]]: MUST be a SIGNATURE (section 2.3.1.12) element containing the cryptographic signature of the body of the certificate, generated by the issuer of the certificate. The hash MUST be the SHA1 hash of the body. The signature MUST be the hash encrypted with the issuer's private key. The key length MUST be the length of the issuer's private key, which MUST match the length of the issuer's public key.

2.3.4.1 DESCRIPTOR

The DESCRIPTOR element of the SPC describes the type of certificate and MUST use the following template.

```
<DESCRIPTOR>
   <OBJECT type="Machine-Certificate">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      <NAME>
         Microsoft Machine-Certificate
      </NAME>
   </OBJECT>
</DESCRIPTOR>
```

[[- GUID -]]: MUST be a unique GUID that identifies the certificate, represented as a literal ASCII string enclosed in braces.

2.3.4.2 ISSUER

The ISSUER element of the SPC identifies the issuer of the certificate. The contents of the ISSUER element MUST be copied verbatim from the contents of the principal element in the ISSUEDPRINCIPALS element of the SPC issuer.

The ISSUER element MUST use the following template.

```
<ISSUER>
   <OBJECT type="[[- type -]]">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      <NAME>
         [[- name -]]
      </NAME>
   </OBJECT>
```

```
    [[- cps -]]
    [[- publickey -]]
</ISSUER>
```

[[- type -]]: MUST be a string that describes the type of the ISSUER. MUST be taken as the object of the principal of the ISSUEDPRINCIPAL of the issuer's certificate. For a version 1 client, this MUST be "MS-DRM-Server". For a version 1 SP1, version 1 SP2, or version 2 client, this MUST be "MS-DRM-Desktop-Security-Processor".

[[- GUID -]]: MUST be a unique GUID that identifies the issuer of the certificate, represented as a literal ASCII string enclosed in braces. MUST be taken from the object of the principal associated with the ISSUEDPRINCIPAL element belonging to the issuer's certificate.

[[- name -]]: MUST be a string that describes the issuer. For a version 1 client, this MUST be "Machine Activation Server". For a version 1 SP1, version 1 SP2, or version 2 client, this MUST be "Microsoft DRM Production Desktop Security Processor Activation Certificate".

[[- cps -]]: SHOULD be present in SPCs for version 1 clients. MUST NOT be present for SPCs for version 1 SP1, version 1 SP2, or version 2 clients. If present, this MUST be a SECURITYLEVEL element with the name "Certificate Practice Statement" and have the value of a URL pointing to a certificate practice statement.

[[- publickey -]]: MUST contain the issuer's public key. Exponent MUST be set to 65537. The size of the issuer's public key MUST be 1,024 bits. The modulus MUST contain the modulus of the issuer's public key.

2.3.4.3 DISTRIBUTIONPOINT

The DISTRIBUTIONPOINT element of the SPC describes the location of the issuer of the SPC.

In the case of a version 1 client, the DISTRIBUTIONPOINT element of the SPC MUST point to the Microsoft RMS Activation cloud service, as shown in the following XML.

```
<DISTRIBUTIONPOINT>
   <OBJECT type="Activation">
      <ID type="MS-GUID">
         {99F48562-703E-4E7D-9175-DD69C66921B7}
      </ID>
      <NAME>
         Microsoft Activation Server
      </NAME>
      <ADDRESS type="URL">
         https://activation.drm.microsoft.com
      </ADDRESS>
   </OBJECT>
</DISTRIBUTIONPOINT>
```

In the case of a version 1 SP1, version 1 SP2, or version 2 client, this refers to the client itself. The element MUST use the following XML.

```
<DISTRIBUTIONPOINT>
   <OBJECT type="Activation">
      <ID type="MS-GUID">
         {99F48562-703E-4E7D-9175-DD69C66921B7}
```

```
        </ID>
        <NAME>
           Microsoft Activation
        </NAME>
        <ADDRESS type="URL">
           file:///rmactivate.exe
        </ADDRESS>
     </OBJECT>
  </DISTRIBUTIONPOINT>
```

2.3.4.4 ISSUEDPRINCIPALS

The ISSUEDPRINCIPALS element of the SPC issues the SPC public key. It MUST use the following template.

```
  <ISSUEDPRINCIPALS>
     <PRINCIPAL>
        <OBJECT type="Machine-Unique-Identifier">
           <ID type="MS-GUID">
              [[- GUID -]]
           </ID>
           <NAME>Machine</NAME>
        </OBJECT>
        [[- publickey -]]
        <DIGEST>
           <ALGORITHM>SHA1</ALGORITHM>
           <PARAMETER name="codingtype">
              <VALUE encoding="string">
                 surface-coding
              </VALUE>
           </PARAMETER>
           <VALUE encoding="base64" size="160">
              [[- hash -]]
           </VALUE>
        </DIGEST>
        [[- platform -]]
        [[- manufacturer -]]
        [[- repository -]]
     </PRINCIPAL>
  </ISSUEDPRINCIPALS>
```

[[- GUID -]]: MUST be a unique GUID that identifies the principal the certificate is issuing, represented as a literal ASCII string enclosed in braces.

[[- publickey -]]: MUST contain the SPC public key. The exponent MUST be set to 65537. For a version 1 client, the size of the SPC public key MUST be 512 bits. For version 1 SP1, version 1 SP2, and version 2 clients, the size of the SPC public key MUST be 1,024 bits. The modulus MUST contain the modulus of the SPC public key.

[[- hash -]]: MUST contain a SHA1 hash of HID information.

[[- platform -]]: MUST contain a SECURITYLEVEL element with the name "Platform" and the value of a string that contains the version of the client platform.

[[- manufacturer -]]: MUST contain a SECURITYLEVEL element with the name "Manufacturer" and the value of a string that contains identifying information about the creator of the security processor.

[[- repository -]]: MUST contain a SECURITYLEVEL element with the name "Repository" and the value of a string that contains the version of the security processor.

2.3.5 RMS Account Certificate

This section defines the format of the RMS account certificate (RAC). The server generates the RAC when it responds to a successful Certify request.

The RAC MUST use the following template.

```
<XrML xmlns="" version="1.2">
   <BODY type="LICENSE" version="3.0">
      [[- issuedtime -]]
      [[- validitytime -]]
      [[- descriptor -]]
      [[- issuer -]]
      [[- distributionpoint-int -]]
      [[- distributionpoint-ext -]]
      [[- issuedprincipals -]]
      [[- federationprincipals -]]
   </BODY>
   [[- signature -]]
</XrML>
```

[[- issuedtime -]]: MUST be an ISSUEDTIME (section 2.3.1.1) element containing the time the RAC was generated, in UTC.

[[- validitytime -]]: SHOULD be a VALIDITYTIME (section 2.3.1.2) element describing the period of validity for the RAC, in UTC.

[[- descriptor -]]: MUST be a DESCRIPTOR (section 2.3.5.1) element describing the RAC.

[[- issuer -]]: MUST be an ISSUER (section 2.3.1.5) element describing the issuer of the RAC.

[[- distributionpoint-int -]]: SHOULD be a DISTRIBUTIONPOINT (section 2.3.1.7) element containing the intranet URL address of the server that issued the RAC.

[[- distributionpoint-ext -]]: SHOULD be a DISTRIBUTIONPOINT element containing the external URL address of the server that issued the RAC.

[[- issuedprincipals -]]: MUST be an ISSUEDPRINCIPALS (section 2.3.1.11) element describing the principal and the RAC public key.

[[- federationprincipals -]]: MUST be a FEDERATIONPRINCIPALS element that issues the RAC private key to the user account.

[[- signature -]]: MUST be a SIGNATURE element containing the cryptographic signature of the body of the certificate, generated by the issuer of the certificate. The hash MUST be the SHA1 hash of the body. The signature MUST be the hash encrypted with the issuer's private key. The key length MUST be the length of the issuer's private key, which MUST match the length of the issuer's public key.

2.3.5.1 DESCRIPTOR

The DESCRIPTOR element of the RAC describes the type of the certificate and MUST use the following template.

```
<DESCRIPTOR>
   <OBJECT type="Group-Identity-Credential">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
   </OBJECT>
</DESCRIPTOR>
```

[[- GUID -]]: MUST be a unique GUID that identifies the certificate, represented as a literal ASCII string enclosed in braces.

2.3.5.2 ISSUER

The ISSUER element of the RAC identifies the issuer of the certificate. The contents of the ISSUER element MUST be copied verbatim from the contents of the principal element in the ISSUEDPRINCIPALS element of the issuing server's SLC.

The ISSUER element MUST use the following template.

```
<ISSUER>
   <OBJECT type="MS-DRM-Server">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      [[- address -]]
   </OBJECT>
   [[- publickey -]]
   [[- serverversion -]]
   [[- serversku -]]
</ISSUER>
```

[[- GUID -]]: MUST be a unique GUID that identifies the issuer of the certificate, represented as a literal ASCII string enclosed in braces. MUST be taken from the object of the principal of the ISSUEDPRINCIPALS of the issuer's certificate.

[[- address -]]: SHOULD be an ADDRESS element of type "URL" containing the URL of the server.

[[- publickey -]]: MUST contain the issuer's public key. The exponent MUST be set to 65537. The size of the issuer's public key MUST be 1,024 bits. The modulus MUST contain the modulus of the issuer's public key.

[[- serverversion -]]: SHOULD be a SECURITYLEVEL element with the name "Server-Version" and the value of a string that contains the version string of the server.

[[- serversku -]]: SHOULD be a SECURITYLEVEL element with the name "Server-SKU" and the value of a string that contains the further version or edition information of the server.

2.3.5.3 DISTRIBUTIONPOINT

The DISTRIBUTIONPOINT elements of the RAC describe the location of the server that issued the RAC and MUST use the following template.

```
<DISTRIBUTIONPOINT>
   <OBJECT type="[[- type -]]">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      <NAME>
         Microsoft Identity Certification Server
      </NAME>
      [[- address -]]
   </OBJECT>
</DISTRIBUTIONPOINT>
```

[[- type -]]: MUST be the type of the DISTRIBUTIONPOINT address. For an intranet address, the type is "Activation". For an external address, the type is "Extranet-Activation".

[[- GUID -]]: MUST be a unique GUID that identifies this DISTRIBUTIONPOINT element, represented as a literal ASCII string enclosed in braces.

[[- address -]]: MUST be an ADDRESS element of type "URL" containing the URL of the server. For an intranet address, this is the internal URL of the server that issued the RAC. For an extranet address, this SHOULD be the external URL of the server that issued the RAC using a **fully qualified domain name (FQDN)**.

2.3.5.4 ISSUEDPRINCIPALS

The ISSUEDPRINCIPALS element of the RAC issues the RAC public key to the user account.

The ISSUEDPRINCIPALS element MUST use the following template.

```
<ISSUEDPRINCIPALS>
   <PRINCIPAL internal-id="1">
      <OBJECT type="Group-Identity">
         <ID type="[[- type -]]">
            [[- userid -]]
         </ID>
         [[- emailaddress -]]
         [[- emailalias -]]
      </OBJECT>
      [[- publickey -]]
      <SECURITYLEVEL name="Group-Identity-Type"
         value="Group" />
         [[- RACtype -]]
      <SECURITYLEVEL name="Group-Identity-Policy"
         value="Group-Identity-Credential" />
   </PRINCIPAL>
</ISSUEDPRINCIPALS>
```

[[- type -]]: MUST be the type of user account, determined by the authentication scheme. For a RAC issued by a server that has authenticated the user by an Active Directory account, the type MUST be "Windows". For a RAC issued by a server using Microsoft Web Browser Federated Sign-On

authentication [MS-MWBF], the type MUST be "Federation". For a RAC issued by the Microsoft RMS Account Certification cloud service using Passport authentication, the type is "Passport".

[[- userid -]]: MUST be the identifier of the user. For a RAC issued to a user's Active Directory credentials, this MUST be the user's security ID (**SID**). For a RAC issued to a user's MWBF credentials, this MUST be a unique GUID. For a RAC issued to a user's Passport credentials, this MUST be the user's Passport User ID (**PUID**).

[[- emailaddress -]]: A NAME element that MUST contain the primary e-mail address associated with the user's account.

[[- emailalias -]]: SHOULD contain an e-mail alias for a Microsoft Web Browser Federated Sign-On authenticated user. MAY exist for RACs of type "Federation". MUST NOT exist for RACs of type "Windows" or "Passport". If present, this MUST be an ADDRESS element of type "email_alias" containing an e-mail address. MAY have multiple elements as peers with one element for each e-mail alias.

[[- publickey -]]: MUST contain the RAC public key. The exponent MUST be set to 65537. The size of the RAC public key MUST be 1,024 bits. The modulus MUST contain the modulus of the RAC public key.

[[- RACtype -]]: MUST describe whether the RAC is considered persistent or temporary. A SECURITYLEVEL element with the name "Group-Identity-Credential-Type" with a value of either "Persistent" or "Temporary".

2.3.5.5 FEDERATIONPRINCIPLES

The FEDERATIONPRINCIPALS element of the RAC issues the RAC private key to the user account and binds it to the machine by encrypting it with the SPC. It MUST use the following template.

```
<FEDERATIONPRINCIPALS>
   <PRINCIPAL>
      [[- machineobject -]]
      [[- enablingbits -]]
      [[- platform -]]
      [[- manufacturer -]]
      [[- repository -]]
   </PRINCIPAL>
</FEDERATIONPRINCIPALS>
```

[[- machineobject -]]: MUST be an object element that identifies the machine. MUST be copied verbatim from the object in the principal element in the ISSUEDPRINCIPALS element of the SPC, including the same GUID.

[[- enablingbits -]]: MUST be the RAC private key encrypted with the SPC public key, contained within an ENABLINGBITS element.

[[- platform -]]: MUST be a SECURITYLEVEL element with the name "Platform" and the value of a string that contains the version of the client platform. MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the SPC.

[[- manufacturer -]]: MUST be a SECURITYLEVEL element with the name "Manufacturer" and the value of a string that contains identifying information about the creator of the security processor. MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the SPC.

[[- repository -]]: MUST be a SECURITYLEVEL element with the name "Repository" and the value of a string that contains the version of the security processor. MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the SPC.

2.3.6 Client Licensor Certificate

This section defines the format of the client licensor certificate (CLC) chain. The server generates the CLC when it responds to a successful **GetClientLicensorCert** request.

The CLC MUST use the following template.

```
<XrML xmlns="" version="1.2">
   <BODY type="LICENSE" version="3.0">
      [[- issuedtime -]]
      [[- descriptor -]]
      [[- issuer -]]
      [[- distributionpoint-int -]]
      [[- distributionpoint-ext -]]
      [[- issuedprincipals -]]
      <WORK>
         [[- workobject -]]
         <RIGHTSGROUP name="Main-Rights">
            <RIGHTSLIST>
               <RIGHT name="ISSUE">
                  <CONDITIONLIST>
                     <TIME>
                        [[- rangetime -]]
                     </TIME>
                     <ACCESS>
                        <PRINCIPAL internal-id="1">
                           [[- enablingbits -]]
                        </PRINCIPAL>
                     </ACCESS>
                  </CONDITIONLIST>
               </RIGHT>
            </RIGHTSLIST>
         </RIGHTSGROUP>
      </WORK>
   </BODY>
   [[- signature -]]
</XrML>
```

[[- issuedtime -]]: MUST be an ISSUEDTIME (section 2.3.1.1) element containing the time the CLC was generated, in UTC.

[[- descriptor -]]: MUST be a DESCRIPTOR (section 2.3.1.4) element describing the CLC.

[[- issuer -]]: MUST be an ISSUER (section 2.3.1.5) element describing the issuer of the CLC.

[[- distributionpoint-int -]]: MUST be a DISTRIBUTIONPOINT (section 2.3.1.7) element containing the intranet URL address of the server that issued the CLC. The server at this address will issue ULs from content that is published using this CLC.

[[- distributionpoint-ext -]]: SHOULD be a DISTRIBUTIONPOINT element containing the external URL address of the server that issued the CLC, but this is optional. The server at this address will issue ULs from content that is published using this CLC.

[[- issuedprincipals -]]: MUST be an ISSUEDPRINCIPALS (section 2.3.1.11) element describing the principal and the CLC public key.

[[- workobject -]]: MUST be an object element that identifies the certificate. Copied verbatim from the object in the DESCRIPTOR (section 2.3.6.1), including the same GUID.

[[- rangetime -]]: MUST be a RANGETIME (section 2.3.1.3) element describing the period during which the certificate can be used for issuance.

[[- enablingbits -]]: MUST be the CLC private key encrypted with the RAC public key, contained within an ENABLINGBITS (section 2.3.1.13) element.

[[- signature -]]: MUST be a SIGNATURE (section 2.3.1.12) element containing the cryptographic signature of the body of the certificate, generated by the issuer of the certificate. The hash MUST be the SHA1 hash of the BODY. The signature MUST be the hash encrypted with the issuer's private key. The key length MUST be the length of the issuer's private key, which MUST match the length of the issuer's public key.

2.3.6.1 DESCRIPTOR

The DESCRIPTOR element of the CLC describes the type of the certificate and MUST use the following template.

```
<DESCRIPTOR>
   <OBJECT type="Client-Licensor-Certificate">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
   </OBJECT>
</DESCRIPTOR>
```

[[- GUID -]]: A unique GUID that identifies the certificate, represented as a literal ASCII string enclosed in braces.

2.3.6.2 ISSUER

The ISSUER element of the CLC identifies the issuer of the certificate. The contents of the ISSUER element MUST be copied verbatim from the contents of the principal element in the ISSUEDPRINCIPALS element of the SLC of the issuing server.

The ISSUER element MUST use the following template.

```
<ISSUER>
   <OBJECT type="MS-DRM-Server">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      [[- address]]
   </OBJECT>
   [[- publickey -]]
   [[- serverversion -]]
   [[- serversku -]]
</ISSUER>
```

[[- GUID -]]: MUST be a unique GUID that identifies the issuer of the certificate, represented as a literal ASCII string enclosed in braces. MUST be taken from the object of the principal of the ISSUEDPRINCIPAL of the issuer's certificate.

[[- address -]]: SHOULD be an ADDRESS element of type "URL" containing the URL of the server.

[[- publickey -]]: MUST contain the issuer's public key. The exponent MUST be set to 65537. The size of the issuer's public key MUST be 1,024 bits. The modulus MUST contain the modulus of the issuer's public key.

[[- serverversion -]]: SHOULD be a SECURITYLEVEL element with the name "Server-Version" and the value of a string that contains the version string of the server.

[[- serversku -]]: SHOULD be a SECURITYLEVEL element with the name "Server-SKU" and the value of a string that contains further version or edition information of the server.

2.3.6.3 DISTRIBUTIONPOINT

The DISTRIBUTIONPOINT elements of the CLC describe the location of the server that issued the CLC. The server at these addresses will be used for issuing ULs from content that is published using this CLC.

The DISTRIBUTIONPOINT elements MUST use the following template.

```
<DISTRIBUTIONPOINT>
   <OBJECT type="[[- type -]]">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      <NAME>
         DRM Server Cluster
      </NAME>
      [[- address -]]
   </OBJECT>
</DISTRIBUTIONPOINT>
```

[[- type -]]: MUST be the type of the DISTRIBUTIONPOINT address. For an intranet address, the type is "License-Acquisition-URL". For an external address, the type is "Extranet-License-Acquisition-URL".

[[- GUID -]]: MUST be a unique GUID that identifies this DISTRIBUTIONPOINT element, represented as a literal ASCII string enclosed in braces.

[[- address -]]: MUST be an ADDRESS element of type "URL" containing the URL of the server. For an intranet address, this is the internal URL of the server that issued the CLC. For an extranet address, this is the external URL of the server that issued the CLC using an FQDN.

2.3.6.4 ISSUEDPRINCIPALS

The ISSUEDPRINCIPALS element of the CLC issues the CLC public key to the user account.

The ISSUEDPRINCIPALS element MUST use the following template.

```
<ISSUEDPRINCIPALS>
   <PRINCIPAL internal-id="1">
```

```
      <OBJECT type="Group-Identity">
         <ID type="[[- type -]]">
            [[- userid -]]
         </ID>
         [[- emailaddress -]]
         [[- emailalias -]]
      </OBJECT>
      [[- publickey -]]
   </PRINCIPAL>
</ISSUEDPRINCIPALS>
```

[[- type -]]: MUST be the type of user account, as determined by the authentication scheme. MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the RAC.

[[- userid -]]: MUST be the identifier of the user. MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the RAC.

[[- emailaddress -]]: MUST be a NAME element that contains the primary e-mail address associated with the user's account.

[[- emailalias -]]: SHOULD contain an e-mail alias for a Microsoft Web Browser Federated Sign-On authenticated user [MS-MWBF]. MAY exist for CLCs issued to RACs of type "Federation". MUST NOT exist for CLCs issued to RACs of type "Windows" or "Passport". If present, this MUST be an ADDRESS element of type "email_alias" containing an e-mail address. MAY have multiple elements as peers with one element for each e-mail alias. MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the RAC.

[[- publickey -]]: MUST contain the CLC public key. The exponent MUST be set to 65537. The size of the CLC public key MUST be 2,048 bits. The modulus MUST contain the modulus of the RAC public key.

2.3.7 Publishing License

This section defines the format of the publishing license (PL). PLs generated from offline publishing are built by the client and signed using the CLC. PLs generated from online publishing are built by the client and signed by the server.

The PL MUST use the following template.

```
<XrML version="1.2" xmlns="">
   <BODY type="Microsoft Rights Label" version="3.0">
      [[- issuedtime -]]
      [[- descriptor -]]
      [[- issuer -]]
      [[- distributionpoint-int -]]
      [[- distributionpoint-ext -]]
      [[- issuedprincipals -]]
      [[- distributionpoint-ref -]]
      <WORK>
         [[- workobject -]]
         <METADATA>
            [[- owner -]]
         </METADATA>
      </WORK>
      [[- authenticateddata -]]
```

```
        <POLICYLIST type="exclusion">
           <POLICY>
              [[- policyobject -]]
           </POLICY>
        </POLICYLIST>
     </BODY>
     [[- signature -]]
  </XrML>
```

[[- issuedtime -]]: MUST be an ISSUEDTIME (section 2.3.1.1) element containing the time the PL was generated, in UTC.

[[- descriptor -]]: MUST be a DESCRIPTOR (section 2.3.1.4) element describing the PL.

[[- issuer -]]: MUST be an ISSUER (section 2.3.1.5) element describing the issuer of the PL.

[[- distributionpoint-int -]]: MUST be a DISTRIBUTIONPOINT (section 2.3.1.7) element containing the intranet URL address of the server that will issue ULs from this PL.

[[- distributionpoint-ext -]]: MUST be a DISTRIBUTIONPOINT element containing the external URL address of the server that will issue ULs from this PL.

[[- issuedprincipals -]]: MUST be an ISSUEDPRINCIPALS (section 2.3.1.11) element describing the principal and the server public key.

[[- distributionpoint-ref -]]: MUST be a DISTRIBUTIONPOINT element containing the author's referral information.

[[- signature -]]: MUST be a SIGNATURE (section 2.3.1.12) element containing the cryptographic signature of the body of the certificate, generated by the issuer of the certificate. The hash MUST be the SHA1 hash of the body. The signature MUST be the hash encrypted with the issuer's private key. The key length MUST be the length of the issuer's private key, which MUST match the length of the issuer's public key.

[[- workobject -]]: MUST be an object element that identifies the content that the PL applies to. This object SHOULD be created by the application used to create the PL and, therefore, SHOULD contain application-specific information.

[[- owner -]]: MUST be an OWNER (section 2.3.7.5) element that describes the author of the document.

[[- authenticateddata -]]: MUST be an AUTHENTICATEDDATA (section 2.3.7.6) element that describes the usage policy issued by the author.

[[- policyobject -]]: MAY be an object element that identifies application policy restrictions that apply to the PL and the information the PL protects, as defined in POLICY (section 2.3.7.7).

2.3.7.1 DESCRIPTOR

The DESCRIPTOR element of the PL describes the type of license and MUST use the following template.

```
<DESCRIPTOR>
   <OBJECT>
      <ID type="MS-GUID">
```

```
            [[- GUID -]]
         </ID>
         [[- name -]]
      </OBJECT>
   </DESCRIPTOR>
```

[[- GUID -]]: MUST be a unique GUID that identifies the license, represented as a literal ASCII string enclosed in braces.

[[- name -]]: MUST be a NAME element giving the name of the policy described in the PL.

2.3.7.2 ISSUER

The ISSUER element of the PL identifies the issuer of the license. The object and PUBLICKEY elements of the ISSUER element MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the CLC for offline publishing.

The object and PUBLICKEY elements of the ISSUER element MUST also be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the SLC by the server for online publishing.

The ISSUER element MUST use the following template.

```
<ISSUER>
   [[- object -]]
   [[- publickey -]]
   <SECURITYLEVEL name="SDK"
                  value="6.0.6000.16386" />
</ISSUER>
```

[[- object -]]: MUST be the object element copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the issuer.

[[- publickey -]]: MUST contain the issuer's public key. The exponent MUST be set to 65537. The size MUST be the size of the issuer's public key in bits. The modulus MUST contain the modulus of the issuer's public key.

2.3.7.3 DISTRIBUTIONPOINT

The DISTRIBUTIONPOINT elements of the PL describe the locations of the server that will be used for issuing ULs based on the PL.

The DISTRIBUTIONPOINT elements MUST use the following template.

```
<DISTRIBUTIONPOINT>
   <OBJECT type="[[- type -]]">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      <NAME>
         [[- name -]]
      </NAME>
      [[- address -]]
   </OBJECT>
```

```
</DISTRIBUTIONPOINT>
```

[[- type -]]: MUST be the type of the DISTRIBUTIONPOINT address. For an intranet address, the type MUST be "License-Acquisition-URL". For an external address, the type MUST be "Extranet-License-Acquisition-URL". For a reference to the author of the document, the type MUST be "Referral-Info".

[[- GUID -]]: MUST be a unique GUID that identifies this DISTRIBUTIONPOINT element, represented as a literal ASCII string enclosed in braces.

[[- address -]]: MUST be an ADDRESS element of type "URL" containing the URL of the server or an e-mail address when the object type is "Referral-Info". For an intranet address, this is the internal URL of the server that issued the PL. For an extranet address, this is the external URL of the server that issued the PL using an FQDN.

[[- name -]]: MUST be a name for the object. For an object of type "Referral-Info", this element MUST contain the literal string "Author". For other objects, this element MUST contain the literal string "DRM Server Cluster".

2.3.7.4 ISSUEDPRINCIPALS

The ISSUEDPRINCIPALS element of the PL issues the content symmetric key to the server.

The ISSUEDPRINCIPALS element MUST use the following template.

```
<ISSUEDPRINCIPALS>
   <PRINCIPAL internal-id="1">
      <OBJECT type="MS-DRM-Server">
         <ID type="MS-GUID">
            [[- GUID -]]
         </ID>
         [[- address -]]
      </OBJECT>
      [[- publickey -]]
      <SECURITYLEVEL name="Server-Version" value="1.0.3246.0" />
      <SECURITYLEVEL name="Server-SKU" value="RMS 1.0" />
      [[- enablingbits -]]
   </PRINCIPAL>
</ISSUEDPRINCIPALS>
```

[[- GUID -]]: MUST be a unique GUID that identifies the server that will issue licenses from this PL, represented as a literal ASCII string enclosed in braces. For an offline-published PL, this MUST be taken from the object of the ISSUER element of the CLC. For an online-published PL, this MUST be taken from the object of the principal of the ISSUEDPRINCIPALS element of the SLC.

[[-address -]]: MUST be an ADDRESS element of type "URL" containing the URL of the server. For an offline-published PL, this MUST be taken from the object of the ISSUER element of the CLC. For an online-published PL, this MUST be taken from the object of the principal of the ISSUEDPRINCIPALS element of the SLC.

[[- publickey -]]: MUST contain the server public key. The exponent MUST be set to 65537. The size MUST be the size of the public key, in bits. The modulus MUST contain the modulus of the server public key. For an offline-published PL, this MUST be taken from the PUBLICKEY of the ISSUER

element of the CLC. For an online-published PL, this MUST be taken from the PUBLICKEY of the principal of the ISSUEDPRINCIPALS element of the SLC.

[[- enablingbits -]]: MUST contain the content symmetric key encrypted with the server public key, contained within an ENABLINGBITS element.

2.3.7.5 OWNER

The OWNER element of the PL describes the author of the PL as a formal principal.

The OWNER element MUST use the following template.

```
<OWNER>
   <OBJECT>
      <ID type=[[- type -]] />
      [[- emailaddress -]]
   </OBJECT>
</OWNER>
```

[[- type -]]: MUST be the type of user account, as determined by the authentication scheme. For an ID authenticated by an Active Directory account, the type MUST be "Windows". For an ID authenticated by a server using the Microsoft Web Browser Federated Sign-On Protocol [MS-MWBF], the type MUST be "Federation". For an ID authenticated by Passport, the type MUST be "Passport".

[[- emailaddress -]]: MUST be a NAME element that contains the primary e-mail address associated with the author's account.

2.3.7.6 AUTHENTICATEDDATA

The AUTHENTICATEDDATA element of the PL MUST contain the usage policy defined by the author of the PL. It MUST be encrypted to the server public key, and the encrypted results MUST be base64-encoded.

The AUTHENTICATEDDATA element MUST use the following template.

```
<AUTHENTICATEDDATA id="Encrypted-Rights-Data">
   [[- encryptedrightsdata -]]
</AUTHENTICATEDDATA>
```

[[- encryptedrightsdata -]]: MUST be the usage policy defined by the author of the PL, encrypted to the server public key, and then base64-encoded. For information on the plaintext description (prior to base64 encoding and encryption), see section 2.3.8.

2.3.7.7 POLICY

The POLICY element of the PL contains usage policy other than user rights. It MAY be used to define application restrictions, such as version requirements of an application that attempts to access the PL. It is created by the application that creates the PL.

If present, the POLICY element MUST use the following template.

```
<POLICY>
   <OBJECT>
```

```
      <ID type="filename">
         [[- filename -]]
      </ID>
      <VERSIONSPAN min="[[- min -]]" max="[[- max -]]" />
   </OBJECT>
</POLICY>
```

[[- filename -]]: MUST be the file name of the application to which the policy applies.

[[- min -]]: MUST be the minimum version of the application named by [*[- filename -]]* that is allowed to access the PL and the protected information.

[[- max-]]: MUST be the maximum version of the application named by *[[- filename -]]* that is allowed to access the PL and the protected information.

2.3.8 Encrypted Rights Data

The contents of the PL's AUTHENTICATEDDATA element having an ID of "Encrypted-Rights-Data" MUST be an XrML document, as defined in [XRML], referred to as Encrypted Rights Data (ERD). The ERD is XrML that defines the rights the author grants. It is encrypted for privacy protection and then base64-encoded. The plaintext ERD MUST use the following template.

```
<XrML xmlns="" version="1.2">
   <BODY type=[[- erdtype -]] >
      [[- issuedtime -]]
      [[- descriptor -]]
      [[- issuer -]]
      [[- distributionpoint-pub -]]
      <WORK>
         <OBJECT>
            <ID type="" />
         </OBJECT>
         <PRECONDITIONLIST>
            <TIME />
         </PRECONDITIONLIST>
         <CONDITIONLIST />
         <RIGHTSGROUP name="Main-Rights">
            <RIGHTSLIST>
               <RIGHT name="OWNER">
                  <CONDITIONLIST>
                     <ACCESS>
                        <PRINCIPAL>
                           <OBJECT>
                              <ID type="Internal">
                                 Owner
                              </ID>
                           </OBJECT>
                        </PRINCIPAL>
                     </ACCESS>
                  </CONDITIONLIST>
               </RIGHT>
               [[- right -]]
            </RIGHTSLIST>
         </RIGHTSGROUP>
      </WORK>
      <AUTHENTICATEDDATA name="VIEWER" id="APPSPECIFIC">
```

```
            1
         </AUTHENTICATEDDATA>
      </BODY>
      [[- signature -]]
   </XrML>
```

[[- erdtype -]]: MUST be the type of ERD. If the ERD was generated based on an enterprise rights template, then this value MUST be "Microsoft Official Rights Template". Otherwise this value MUST be "Microsoft Rights Template".

[[- issuedtime -]]: MUST be an ISSUEDTIME (section 2.3.1.1) element containing the time the ERD was generated, in UTC.

[[- descriptor -]]: MUST be a DESCRIPTOR (section 2.3.1.4) element describing the ERD.

[[- issuer -]]: MUST be an ISSUER (section 2.3.1.5) element describing the issuer of the ERD.

[[- distributionpoint-pub -]]: MUST be a DISTRIBUTIONPOINT (section 2.3.1.7) element containing the URL address of the server that will issue ULs for this ERD.

[[- right -]]: MUST be an element that defines a right and the principal that possesses the right, as defined in section 2.3.8.4.

2.3.8.1 DESCRIPTOR

The DESCRIPTOR element of the ERD describes the ERD and MUST use the following template.

```
<DESCRIPTOR>
   <OBJECT>
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      [[- name -]]
   </OBJECT>
</DESCRIPTOR>
```

[[- GUID -]]: MUST be a unique GUID that identifies this DISTRIBUTIONPOINT element, represented as a literal ASCII string enclosed in braces.

[[- name -]]: MUST be a NAME element providing the name of the policy described in the ERD. The text of this element is structured as follows, and one or more occurrences of the following structure MUST be present in each ERD descriptor, separated by a colon.

```
LCID [[- lcid -]]:NAME [[- name2 -]]:DESCRIPTION [[- description -]];
```

[[- lcid -]]: MUST be the locale identifier (LCID) describing the language in which the name and description that follow it are encoded.

[[- name2 -]]: MUST be the name of the policy, encoded in the language defined by the *[[- lcid -]]*.

[[- description -]]: MUST be the description of the policy, encoded in the language defined by the *[[- lcid -]]*.

2.3.8.2 ISSUER

The ISSUER element of the ERD MUST identify the issuer of the ERD. The object and PUBLICKEY elements of the ISSUER element MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the template if it is based on a template.

The object and PUBLICKEY elements of the ISSUER element MUST be copied verbatim from the PRINCIPAL element in the ISSUEDPRINCIPALS element of the CLC if a template is not used.

The ISSUER element MUST use the following template.

```
<ISSUER>
   [[- object -]]
   [[- publickey -]]
</ISSUER>
```

[[- object -]]: MUST be an object element copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the issuer.

[[- publickey -]]: MUST contain the issuer's public key. The exponent MUST be set to 65537. The size MUST be the size of the issuer's public key, in bits. The modulus MUST contain the modulus of the issuer's public key.

2.3.8.3 DISTRIBUTIONPOINT

The DISTRIBUTIONPOINT element of the ERD describes the location of the server that will be used for issuing ULs based on the ERD. The DISTRIBUTIONPOINT elements MUST use the following template.

```
<DISTRIBUTIONPOINT>
   <OBJECT type="[[- type -]]">
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      <NAME>
         [[- name -]]
      </NAME>
      [[- address -]]
   </OBJECT>
</DISTRIBUTIONPOINT>
```

[[- type -]]: MUST be the type of the DISTRIBUTIONPOINT address. For an ERD the type is "Publishing-URL".

[[- GUID -]]: MUST be a unique GUID that identifies this DISTRIBUTIONPOINT element, represented as a literal ASCII string enclosed in braces.

[[- address -]]: MUST be an ADDRESS element of type "URL" containing the URL of the server.

[[- name -]]: MUST be a name for the object. For an object of type "Publishing-URL", this element contains the text "Publishing Point".

2.3.8.4 RIGHT

The RIGHT element describes a right assigned to a principal. One or more RIGHT elements MUST be present. The RIGHT element MUST follow one of the two following forms.

Form 1

```
<RIGHT name=[[- rightname -]] >
   <CONDITIONLIST>
      [[- timecondition -]]
      <ACCESS>
         <PRINCIPAL>
            <OBJECT>
               <ID type="Unspecified" />
               [[- emailaddress -]]
            </OBJECT>
         </PRINCIPAL>
      </ACCESS>
   </CONDITIONLIST>
</RIGHT>
```

Form 2

```
<[[- rightname -]] >
   <CONDITIONLIST>
      [[- timecondition -]]
      <ACCESS>
         <PRINCIPAL>
            <OBJECT>
               <ID type="Unspecified" />
               [[- emailaddress -]]
            </OBJECT>
         </PRINCIPAL>
      </ACCESS>
   </CONDITIONLIST>
</[[- rightname -]] >
```

[[- rightname -]]: In form 1, the name of the right MUST be an attribute on a RIGHT element and can be any arbitrary right name. In form 2, the name of the right MUST be the name of the element, and MUST be one of a set of the following reserved rights:

- VIEW
- PRINT
- EDIT
- FORWARD
- VIEWRIGHTSDATA

[[- timecondition -]]: MAY exist to specify a number of days for which the right may be exercised. If present, this MUST take the following form:

```
<TIME>
   <INTERVALTIME days="[[- intervaltime -]]" />
</TIME>
```

[[- intervaltime -]]: MUST be the number of days specified for the time condition.

[[- emailaddress -]]: MUST be a NAME element that contains the primary e-mail address associated with the user's account that possesses the right.

2.3.9 Use License

This section defines the format of the use license (UL). The UL names an issued principal via the ISSUEDPRINCIPALS element and then grants a set of rights to that principal, one right per RIGHT element.

The UL MUST use the following template.

```
<XrML version="1.2" xmlns="" purpose="Content-License">
   <BODY type="LICENSE" version="3.0">
      [[- issuedtime -]]
      [[- descriptor -]]
      [[- issuer -]]
      [[- issuedprincipals -]]
      <WORK>
        [[- workobject -]]
        <METADATA>
           [[- owner -]]
        </METADATA>
        <RIGHTSGROUP name="Main-Rights">
           <RIGHTSLIST>
              [[- right -]]
           </RIGHTSLIST>
        </RIGHTSGROUP>
      </WORK>
      <CONDITIONLIST>
         <REFRESH>
           <DISTRIBUTIONPOINT>
              <OBJECT type="Revocation">
                 <ID type="ascii-tag">
                    Irrevocable
                 </ID>
              </OBJECT>
           </DISTRIBUTIONPOINT>
           <INTERVALTIME />
         </REFRESH>
      </CONDITIONLIST>
      <POLICYLIST type="exclusion">
          <POLICY>
             [[- policyobject -]]
          </POLICY>
       </POLICYLIST>
   </BODY>
   [[- signature -]]
</XrML>
```

[[- issuedtime -]]: MUST be an ISSUEDTIME (section 2.3.1.1) element containing the time the UL was generated, in UTC.

[[- descriptor -]]: MUST be a DESCRIPTOR (section 2.3.1.4) element describing the UL.

[[- issuer -]]: MUST be an ISSUER (section 2.3.1.5) element describing the issuer of the UL.

[[- issuedprincipals -]]: MUST be an ISSUEDPRINCIPALS (section 2.3.1.11) element describing the principal and the user public key for which the UL is issued.

[[- workobject -]]: MUST be an object element that identifies the content to which the UL applies. This object is created by the application used to create the PL that the UL was generated from, and therefore will contain application-specific information.

[[- owner -]]: MUST be an OWNER (section 2.3.9.4) element that describes the author of the document.

[[- right -]]: MUST be an element, as defined in section 2.3.9.5, that defines a right and the principal that possesses the right.

[[- policyobject -]]: MAY be an object element that identifies application policy restrictions that apply to the UL and the information the UL protects, as defined in section 2.3.9.6.

[[- signature -]]: MUST be a SIGNATURE (section 2.3.1.12) element containing the cryptographic signature of the body of the certificate, generated by the issuer of the certificate. The hash MUST be the SHA1 hash of the body. The signature MUST be the hash encrypted with the issuer's private key. The key length MUST be the length of the issuer's private key, which MUST match the length of the issuer's public key.

2.3.9.1 DESCRIPTOR

The DESCRIPTOR element of the UL describes the UL and MUST use the following template.

```
<DESCRIPTOR>
   <OBJECT>
      <ID type="MS-GUID">
         [[- GUID -]]
      </ID>
      [[- name -]]
   </OBJECT>
</DESCRIPTOR>
```

[[- GUID -]]: MUST be a unique GUID that identifies this DISTRIBUTIONPOINT element, represented as a literal ASCII string enclosed in braces.

[[- name -]]: MUST be a NAME element giving the name of the policy described in the UL.

2.3.9.2 ISSUER

The ISSUER element of the UL identifies the issuer of the license. The object and PUBLICKEY elements of the ISSUER element MUST be copied verbatim from the object and PUBLICKEY elements of the ISSUER element in the PL used to generate this UL.

The ISSUER element MUST use the following template.

```
<ISSUER>
   [[- object -]]
   [[- publickey -]]
</ISSUER>
```

[[- object -]]: MUST be an object element copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the issuer.

[[- publickey -]]: MUST contain the issuer's public key. The exponent MUST be set to 65537. The size MUST be the size of the issuer's public key, in bits. The modulus MUST contain the modulus of the issuer's public key.

2.3.9.3 ISSUEDPRINCIPALS

The ISSUEDPRINCIPALS element of the UL identifies the RAC to which this UL is issued. All rights in the UL are granted to this RAC. The principal element MUST be copied verbatim from the principal element in the ISSUEDPRINCIPALS element of the RAC.

The ISSUEDPRINCIPALS element MUST use the following template.

```
<ISSUEDPRINCIPALS>
   <PRINCIPAL internal-id="1">
      <OBJECT type="Group-Identity">
         <ID type="[[- type -]]">[[- userid -]]</ID>
         [[- emailaddress -]]
         [[- emailalias -]]
      </OBJECT>
      [[- publickey -]]
      <SECURITYLEVEL name="Group-Identity-Type"
                     value="Group" />
      [[- RACtype -]]
      <SECURITYLEVEL name="Group-Identity-Policy"
                     value="Group-Identity-Credential" />
   </PRINCIPAL>
</ISSUEDPRINCIPALS>
```

[[- type -]]: MUST be the type of user account, determined by the authentication scheme. For a RAC issued by a server that has authenticated the user by an Active Directory account, the type MUST be "Windows". For a RAC issued by a server using Microsoft Web Browser Federated Sign-On authentication [MS-MWBF], the type MUST be "Federation". For a RAC issued by the Microsoft RMS Account Certification cloud service using Passport authentication, the type is "Passport".

[[- userid -]]: MUST be the identity of the user. For a RAC issued to a user's Active Directory credentials, this MUST be the user's SID. For a RAC issued to a user's Microsoft Web Browser Federated Sign-On credentials, this MUST be a unique GUID. For a RAC issued to a user's Passport credentials, this MUST be the user's PUID.

[[- emailaddress -]]: MUST be a NAME element that contains the primary e-mail address associated with the user's account.

[[- emailalias -]]: Contains an e-mail alias for a Microsoft Web Browser Federated Sign-On authenticated user [MS-MWBF]. This element MAY exist for RACs of type "Federation". This element MUST NOT exist for RACs of type "Windows" or "Passport". If present, this MUST be an ADDRESS

element of type "email_alias" containing an e-mail address. There MAY be multiple ADDRESS elements as peers with one element for each e-mail alias.

[[- publickey -]]: MUST contain the RAC public key. The exponent is set to 65537. The size MUST be the size of the RAC public key, in bits. The modulus MUST contain the modulus of the RAC public key.

[[- RACtype -]]: MUST describe whether the RAC is considered persistent or temporary. A SECURITYLEVEL element with the name "Group-Identity-Credential-Type" with a value of either "Persistent" or "Temporary".

2.3.9.4 OWNER

The OWNER element of the UL describes the author of the PL that was used to create the UL. It grants no rights by itself, whereas the RIGHT element with name OWNER does formally grant the owner rights.

The OWNER element MUST follow this template.

```
<OWNER>
   <OBJECT>
      <ID type=[[- type -]] />
      [[- emailalias -]]
   </OBJECT>
</OWNER>
```

[[- type -]]: MUST be the type of user account, as determined by the authentication scheme. For an ID authenticated by an Active Directory account, the type MUST be "Windows". For an ID authenticated by a server using the Microsoft Web Browser Federated Sign-On Protocol [MS-MWBF], the type MUST be "Federation". For an ID authenticated by Passport, the type MUST be "Passport".

[[- emailalias -]]: MUST be a NAME element that contains the primary e-mail address associated with the user's account.

2.3.9.5 RIGHT

The RIGHT element describes a right assigned to the principal named in the use license. One or more RIGHT elements MUST be present.

Each RIGHT element MUST use one of the two following template forms.

Form 1

```
<RIGHT name=[[- rightname -]] >
   <CONDITIONLIST>
      <ACCESS>
         <PRINCIPAL internal-id="1">
            [[- enablingbits -]]
         </PRINCIPAL>
      </ACCESS>
   </CONDITIONLIST>
</RIGHT>
```

Form 2

```
<[[- rightname -]] >
   <CONDITIONLIST>
      <ACCESS>
         <PRINCIPAL internal-id="1">
            [[- enablingbits -]]
         </PRINCIPAL>
      </ACCESS>
   </CONDITIONLIST>
</[[- rightname -]] >
```

[[- rightname -]]: In form 1, the name of the right MUST be a name attribute on a RIGHT element and can be any arbitrary right name. In form 2, the name of the right MUST be the name of the element and MUST be one of a set of the following reserved rights:

- VIEW
- PRINT
- EDIT
- FORWARD
- VIEWRIGHTSDATA
- OWNER

If the UL has been issued to the author of the original PL, then there MUST be one RIGHT element named OWNER and it MUST follow form 1. All rights to the protected information are granted to this owner and further RIGHT elements MUST NOT be present.

[[- enablingbits -]]: MUST contain the content symmetric key encrypted with the user's public key, contained within an ENABLINGBITS element.

2.3.9.6 POLICY

The POLICY element of the UL contains usage policy other than user rights. It MUST be copied verbatim from the PL, if present. It MAY be used to define application restrictions, such as version requirements of an application that tries to access the PL. It is created by the application that creates the PL.

The POLICY element MUST use the following template.

```
<POLICY>
   <OBJECT>
      <ID type="filename">[[- filename -]]</ID>
      <VERSIONSPAN min="[[- min -]]" max="[[- max -]]" />
   </OBJECT>
</POLICY>
```

[[- filename -]]: MUST be the file name of the application to which the policy applies.

[[- min -]]: MUST be the minimum version of the application named by *[[- filename -]]* that is allowed to access the UL and the protected information.

[[- max-]]: MUST be the maximum version of the application named by *[[- filename -]]* that is allowed to access the UL and the protected information.

2.3.10 Rights Policy Template

This section defines the format of the rights policy template. Templates are generated by an administrator on the server and then distributed to client machines. A client generates a PL from a template when a user uses it to protect a document (offline publishing). The PL is signed using the CLC.

The rights policy template MUST use the following template.

```
<XrML version="1.2" xmlns="">
  <BODY type="Microsoft Official Rights Template">
    [[- issuedtime -]]
    [[- descriptor -]]
    [[- issuer -]]
    [[- distributionpoint-pub -]]
     [[- distributionpoint-ref -]]
    [[- work -]]
    [[- authenticateddata -]]
</BODY>
[[- signature -]]
</XrML>
```

[[- issuedtime -]]: MUST be an ISSUEDTIME element containing the time the rights policy template was generated, in UTC.

[[- descriptor -]]: MUST be a DESCRIPTOR element describing the rights policy template, as defined in section 2.3.10.1.

[[- issuer -]]: MUST be an ISSUER element describing the issuer of the rights policy template, as defined in section 2.3.10.2.

[[- distributionpoint-pub -]]: MUST be a DISTRIBUTIONPOINT element containing the intranet licensing URL of the server that will issue ULs for the PL generated from this rights policy template, as specified in section 2.3.10.3.

[[- distributionpoint-ref -]]: MUST be a DISTRIBUTIONPOINT element containing the rights request referral information, as specified in section 2.3.10.3.

[[-work -]]: MUST be a WORK element containing the policy, as specified in section 2.3.10.4.

[[- authenticateddata -]]: MUST be an AUTHENTICATEDDATA element that describes the usage policy issued by the author, as specified in section 2.3.10.5.

[[- signature -]]: MUST be a SIGNATURE element containing the cryptographic signature of the body of the certificate, generated by the issuer of the certificate. The hash MUST be the SHA1 hash of the body. The signature MUST be the hash encrypted with the issuer's private key. The key length MUST be the length of the issuer's private key, which MUST match the length of the issuer's public key.

2.3.10.1 DESCRIPTOR

The DESCRIPTOR element of the rights policy template describes the type of the license and MUST use the following template.

```
<DESCRIPTOR>
  <OBJECT>
    <ID type="MS-GUID">[[- GUID -]]</ID>
    [[- name -]]
  </OBJECT>
</DESCRIPTOR>
```

[[- GUID -]]: MUST be a unique GUID that identifies the rights policy template, represented as a literal ASCII string enclosed in braces.

[[- name -]]: MUST be a NAME element providing the name of the rights policy template. The text of this element is structured as follows, and one or more occurrences of the following structure MUST be present in each NAME element, separated by a colon:

```
LCID [[- lcid -]]:NAME [[- name2 -]]:DESCRIPTION [[- description -]];
```

[[- lcid -]]: MUST be the LCID describing the language in which the *NAME* and *DESCRIPTION* that follow it are encoded.

[[- name2 -]]: MUST be the name of the policy, encoded in the language defined by the [[- lcid -]].

[[- description -]]: MUST be the description of the policy, encoded in the language defined by the [[- lcid -]].

2.3.10.2 ISSUER

The ISSUER element of the rights policy template identifies the issuer of the template. The contents of the ISSUER element MUST be copied from the contents of the principal element in the ISSUEDPRINCIPALS element of the SPC of the issuing server.

The ISSUER element MUST use the following template.

```
<ISSUER>
  <OBJECT type="MS-DRM-Server">
    <ID type="MS-GUID">[[- GUID -]]</ID>
    [[- name -]]
    [[- address -]]
  </OBJECT>
  [[- publickey -]]
</ISSUER>
```

[[- GUID -]]: MUST be a unique GUID that identifies the issuer of the license, represented as a literal ASCII string enclosed in braces. MUST be taken from the object of the principal of the ISSUEDPRINCIPALS of the issuer's certificate.

[[- name -]]: SHOULD be a string containing a name for the server. The NAME element MAY be omitted.

[[- address -]]: SHOULD be an ADDRESS element of type "URL" containing the URL of the server.

[[- publickey -]]: MUST contain the issuer's public key. The exponent MUST be set to 65537. The size of the issuer's public key MUST be 1,024 bits. The modulus MUST contain the modulus of the issuer's public key.

2.3.10.3 DISTRIBUTIONPOINT

The DISTRIBUTIONPOINT element of the rights policy template either describes the intranet licensing URL of the server that will be used for issuing ULs for the PL generated from the rights policy template (publishing point element), or the URL that is used when a recipient of a protected document wants to request rights to the document (referral-info element). If the element describes the location of the server, it can be either an internal or an external location.

The DISTRIBUTIONPOINT elements MUST use the following template.

```
<DISTRIBUTIONPOINT>
  <OBJECT type="[[- type -]]">
    <ID type="MS-GUID">[[- GUID -]]</ID>
    [[- name -]]
    [[- address -]]
  </OBJECT>
</DISTRIBUTIONPOINT>
```

[[- type -]]: MUST be the type of the DISTRIBUTIONPOINT address. For the publishing point element, the type is "Publishing-URL", and for the referral-info element, the type is "Referral-Info".

[[- GUID -]]: MUST be a unique GUID that identifies the DISTRIBUTIONPOINT element, represented as a literal ASCII string enclosed in braces.

[[- name -]]: MUST be a name for the object. For an object of type "Publishing-URL", this element MUST contain the text "Publishing Point", while for an object of type "referral-info", this MUST NOT be present.

[[- address -]]: MUST be an ADDRESS element of type "URL". For an object of type "Publishing-URL", this element MUST contain the intranet licensing URL of the server, while for an object of type "referral-info", this element MUST contain the URL to use for requesting rights (usually an e-mail address).

2.3.10.4 WORK

The WORK element MUST use the following template.

```
<WORK>
      <OBJECT>
        <ID type="" />
      </OBJECT>
[[- preconditionlist -]]
      <RIGHTSGROUP name="Main-Rights">
        <RIGHTSLIST>
          <RIGHT name="OWNER">
            <CONDITIONLIST>
```

```
            <ACCESS>
              <PRINCIPAL>
                <OBJECT>
                  <ID type="Internal">Owner</ID>
                </OBJECT>
              </PRINCIPAL>
            </ACCESS>
          </CONDITIONLIST>
        </RIGHT>
        [[- right -]]
      </RIGHTSLIST>
    </RIGHTSGROUP>
  </WORK>
```

[[- preconditionlist -]]: This element specifies the time conditions on the usage policy, as specified in section 2.3.10.4.1.

2.3.10.4.1 PRECONDITIONLIST

The PRECONDITIONLIST element specifies the period of time for which the document may be accessed. The element MAY be present.

The element MAY be specified in two ways. One of the following two ways MUST be used if this element is present.

Method 1

```
<TIME
  <RANGETIME>
    <FROM>[[- fromtime -]]</FROM>
    <UNTIL>[[- untiltime -]]</UNTIL>
  </RANGETIME>
</TIME>
```

[[- fromtime -]]: The *fromtime* element specifies the beginning date and time for the document to be considered valid (as in "not expired"). The time is expressed in UTC format.

[[- untiltime -]]: The *untiltime* element specifies the end date and time for the document to be considered valid (as in "not expired"). The time is expressed in UTC format.

Method 2

```
<TIME
  <INTERVALTIME days="[[= numberofdays -]]"/>
</TIME>
```

[[- numberofdays -]]: The *numberofdays* element specifies the number of days from the ISSUEDTIME that the document will be considered valid (as in "not expired").

2.3.10.4.2 RIGHTSGROUP

The RIGHTSGROUP element contains RIGHT elements and users who have each of these rights.

2.3.10.4.2.1 RIGHT

The RIGHT element describes a right that is assigned to a principal. One or more RIGHT elements MUST be present. It MUST follow one of two forms.

Form 1

```
<RIGHT name=[[- rightname -]] >
  <CONDITIONLIST>
    [[- timecondition -]]
    <ACCESS>
      <PRINCIPAL>
        <OBJECT>
          <ID type="Unspecified" />
          [[- emailaddress -]]
        </OBJECT>
      </PRINCIPAL>
    </ACCESS>
  </CONDITIONLIST>
</RIGHT>
```

Form 2

```
<[[- rightname -]] >
   <CONDITIONLIST>
      [[- timecondition -]]
       <ACCESS>
         <PRINCIPAL>
            <OBJECT>
               <ID type="Unspecified" />
                  [[- emailaddress -]]
               </OBJECT>
         </PRINCIPAL>
      </ACCESS>
   </CONDITIONLIST>
</[[- rightname -]] >
```

[[- rightname -]]: In form 1, the name of the RIGHT MUST be an attribute on a RIGHT element and can be any arbitrary RIGHT name. In form 2, the name of the RIGHT MUST be the name of the element and MUST be one of a set of the following reserved values:

- VIEW
- PRINT
- EDIT
- FORWARD
- VIEWRIGHTSDATA

[[- timecondition -]]: MAY exist to specify a number of days for which the right may be exercised. If present, this MUST take the following form:

```
<TIME>
```

```
    <INTERVALTIME days="[[- intervaltime -]]" />
</TIME>
```

[[- intervaltime -]]: MUST be the number of days specified for the time condition.

[[- emailaddress -]]: MUST be a NAME element that contains the primary e-mail address associated with the user's account that possesses the right.

2.3.10.5 AUTHENTICATEDDATA

The AUTHENTICATEDDATA element of the template contains the usage policy defined by the rights policy template author. For a template, this element always represents application-specific data. One or more AUTHENTICATEDDATA elements MAY be present and MUST use the following forms.

The AUTHENTICATEDDATA element MUST use the following template.

```
<AUTHENTICATEDDATA name="[[- name - ]]" id="APPSPECIFIC">[[- value -]]
</AUTHENTICATEDDATA>
```

[[- name -]]: The name of the application-specific control.

There are two predefined controls:

VIEWER: Specifies whether the protected document can be opened in a browser.

NOLICCACHE: Specifies whether the use license received from the server should be cached (stored in the client's local store).

[[- value -]]: The value of the application-specific control. For the predefined controls above, the value indicates the following:

VIEWER: '0': Do not allow viewing in a browser; '1': Allow viewing in a browser.

NOLICCACHE: '0': Allow UL caching, '1': Do not allow UL caching.

3 Protocol Details

The following sections specify details of the RMS: Client-to-Server Protocol:

The RMS: Client-to-Server Protocol operates between a client (the initiator) and a server (the responder), and acts as either a creator or a consumer. After server bootstrapping, the protocol allows for stateless server operation. The server MAY retain state where appropriate as an optimization.<8>

3.1 Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The organization is provided to explain how the protocol behaves. This specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this specification.

Trusted security processor CA key: A key that represents an issuer of SPCs. The server SHOULD maintain a list of trusted SPC issuer keys so that it can determine whether to authorize client requests that involve a given SPC chain. The SPC issuer key can be retrieved from the SPC chain.<9>

Server key pair: An asymmetric key pair used for encryption and signing in to the server.<10>

SLC chain: An XrML 1.2 certificate chain that signs the RMS server's public key into the Microsoft certificate hierarchy.

Note that the above conceptual data can be implemented by using a variety of techniques. Any data structure that stores the above conceptual data MAY be used in the implementation.

3.1.2 Timers

This protocol has no timers.

3.1.3 Initialization

The following initialization steps MUST be performed:

1. If the server key pair does not exist, a new server key pair MUST be generated and stored.

2. If an SLC chain does not exist, a new SLC chain MUST be acquired.

3.1.3.1 Acquiring an SLC Chain

A server MUST<11> have an SLC chain that contains its unique public key, grants the server the right to issue certificates and licenses, and leads back to the common RMS root. Microsoft operates a publicly available RMS enrollment cloud service that signs an unsigned SLC and returns an SLC chain that leads back to the common RMS root. The service is open to all callers, performs no authentication and no authorization, and does not require the caller to meet any requirements. Microsoft retains no data.

This service is available for both synchronous and asynchronous requests. The server MUST send information about itself, such as its public key and GUID, to the cloud service. The cloud service

uses this information to generate an SLC, sign it with its private key, append its own certificate chain, and return the result to the server.

- Synchronous: https://activation.drm.microsoft.com/enrollment/enrollservice.asmx
- Asynchronous: https://activation.drm.microsoft.com/offlineenroll/Enrollment.aspx

3.1.3.2 Synchronous Enrollment

The enrollment cloud service uses a SOAP over HTTP protocol, as specified in [SOAP 1.1].

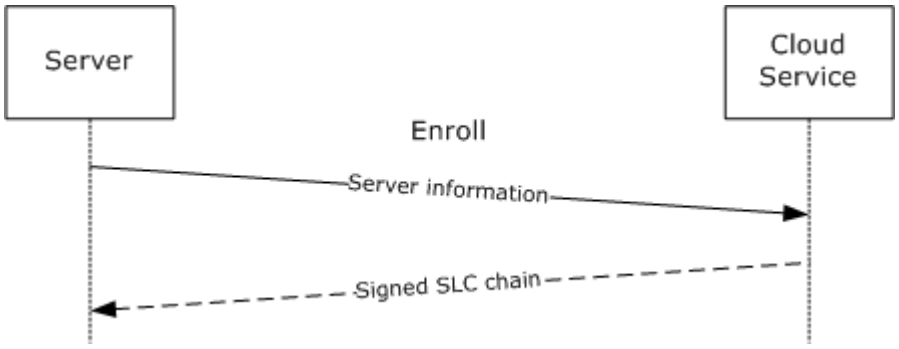

Figure 6: Enrollment message sequence

In the enrollment protocol, the server makes an Enroll request submitting information about itself, including its public key, its unique GUID, the type of revocation to use, and SKU and version information about the server. The cloud service generates the SIGNATURE element of the SLC using its private key, appends the element to the SLC, and appends its own certificate chain. It then returns the signed SLC chain to the server in the response.

3.1.3.2.1 Enroll Request

The Enroll request MUST use the following template.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData
         xmlns="http://microsoft.com/DRM/EnrollmentService">
         <MinimumVersion>[[- minimumversion -]]</MinimumVersion>
         <MaximumVersion>[[- maximumversion -]]</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <Enroll xmlns="http://microsoft.com/DRM/EnrollmentService">
         <oInput>
            <AuthorizationInformation>
               <SignedDataBase64Encoded></SignedDataBase64Encoded>
            </AuthorizationInformation>
            <RevocationInformation>
               <RevocationType>
                  [[- revocationtype -]]
```

```
                </RevocationType>
                <aRevocationAuthorities>
                   [[- revocationauthorities -]]
                </aRevocationAuthorities>
             </RevocationInformation>
             <CertificatePublicKey>
                <aPublicKeyBytes>
                   [[- publickeystring -]]
                </aPublicKeyBytes>
                <Guid>[[- GUID -]]</Guid>
             </CertificatePublicKey>
             <EnrolleeInformation>
                <SKU>[[- serversku -]]</SKU>
                <Version>[[- serverversion -]]</Version>
                <Name>[[- name -]]</Name>
                <URL>[[- url -]]</URL>
             </EnrolleeInformation>
          </oInput>
       </Enroll>
    </soap:Body>
 </soap:Envelope>
```

[[- minimumversion -]]: The minimum version of the enrollment service, specified as a string. MUST be "1.0.0.0".

[[- maximumversion -]]: The maximum version of the enrollment service, specified as a string. MUST be "1.0.0.0".

[[- revocationtype -]]: MUST be either "StandardRevocation" or "CustomRevocation", specified as a string. Although "NonRevocable" is specified as a possible value by WSDL, it is not supported. "StandardRevocation" indicates that the issuer can revoke the SLC. "CustomRevocation" indicates that a third party specified by *[[- revocationauthorities -]]* can revoke the SLC. "StandardRevocation" is recommended.

[[- revocationauthorities -]]: MUST exist only if RevocationType is set to "CustomRevocation" and MUST be empty otherwise. If RevocationType is set to "CustomRevocation", this MUST contain one or more RevocationAuthorityInformation elements.

[[- publickeystring -]]: MUST contain the server's RSA PKCS#1-encoded public key as a base64-encoded string.

[[- GUID -]]: MUST be a unique GUID that identifies the server, represented as a literal ASCII string. MAY be enclosed in braces.

[[- serversku -]]: MUST be a string containing SKU or edition information for the server.

[[- serverversion -]]: MUST be a string containing version information for the server.

[[- name -]]: SHOULD be a string containing a name for the server. The NAME element MAY be omitted.

[[- url -]]: MUST be a string containing a URL for the server.

3.1.3.2.1.1 RevocationAuthorityInformation

If the Enroll request specifies CustomRevocation, at least one RevocationAuthorityInformation element MUST be present to describe the public key of a third-party revocation authority that is allowed to revoke the SLC. A RevocationAuthorityInformation element MUST use the following template.

```
<RevocationAuthorityInformation>
   <aRevocationAuthorityPublicKey>
      [[- key -]]
   </aRevocationAuthorityPublicKey>
</RevocationAuthorityInformation>
```

[[- key -]]: MUST contain the revocation authority's RSA PKCS#1-encoded public key as a base64-encoded string. If this revocation authority is required to issue a revocation list that revokes the SLC, it MUST be issued using this public key and signed with the corresponding private key.

3.1.3.2.2 Enroll Response

The Enroll response MUST use the following template.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData
         xmlns="http://microsoft.com/DRM/EnrollmentService">
         <MinimumVersion>
            [[- minimumversion -]]
         </MinimumVersion>
         <MaximumVersion>
            [[- maximumversion -]]
         </MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <EnrollResponse
         xmlns="http://microsoft.com/DRM/EnrollmentService">
         <EnrollResult>
            <LicensorCertificateChain>
               <string>[[- SLC -]]</string>
               <string>[[- EnrollmentServiceCert -]]</string>
               <string>[[- EnrollmentCACert -]]</string>
               <string>[[- CACert -]]</string>
            </LicensorCertificateChain>
         </EnrollResult>
      </EnrollResponse>
   </soap:Body>
</soap:Envelope>
```

[[- minimumversion -]]: The minimum version of the enrollment service, specified as a string. MUST be "1.0.0.0".

[[- maximumversion -]]: The maximum version of the enrollment service, specified as a string. MUST be "1.0.0.0".

[[- SLC -]]: MUST be a string containing the SLC.

[[- EnrollmentServiceCert -]]: MUST be a string containing the Enrollment Service certificate.

[[- EnrollmentCACert -]]: MUST be a string containing the Enrollment CA certificate.

[[- CACert -]]: MUST be a string containing the CA certificate.

3.1.3.3 Asynchronous Enrollment

To enable "airgap" networks that do not have Internet connectivity, the Enroll SOAP request can be written to an ASCII text file and submitted asynchronously at https://activation.drm.microsoft.com/offlineenroll/Enrollment.aspx.

3.1.3.3.1 Request

The request MUST be an ASCII text file that adheres to the following template.

```
<?xml version="1.0"?>
<EnrollParameters xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <RevocationInformation
      xmlns="http://microsoft.com/DRM/EnrollmentService">
      <RevocationType>
         [[- revocationtype -]]
      </RevocationType>
      <aRevocationAuthorities>
         [[- revocationauthorities -]]
      </aRevocationAuthorities>
   </RevocationInformation>
   <CertificatePublicKey
      xmlns="http://microsoft.com/DRM/EnrollmentService">
      <aPublicKeyBytes>
         [[- publickeystring -]]
      </aPublicKeyBytes>
      <Guid>[[- GUID -]]</Guid>
   </CertificatePublicKey>
   <EnrolleeInformation
      xmlns="http://microsoft.com/DRM/EnrollmentService">
      <SKU>[[[- serversku -]]</SKU>
      <Version>[[- serverversion -]]</Version>
      <URL>[[- url -]]</URL>
   </EnrolleeInformation>
</EnrollParameters>
```

[[- revocationtype -]]: MUST be either "StandardRevocation" or "CustomRevocation", specified as a string. "StandardRevocation" indicates that the issuer can revoke the SLC. "CustomRevocation" indicates that a third party specified by *[[- revocationauthorities -]]* can revoke the SLC. "StandardRevocation" is recommended.

[[-revocationauthorities -]]: MUST exist only if RevocationType is set to "CustomRevocation" and MUST be empty otherwise. If RevocationType is set to "CustomRevocation", this MUST contain one or more RevocationAuthorityInformation elements, as specified in section 3.1.3.2.1.1.

[[- publickeystring -]]: MUST contain the server's RSA PKCS#1-encoded public key as a base64-encoded string.

[[- GUID -]]: MUST be a unique GUID that identifies the server, represented as a literal ASCII string. MAY be enclosed in braces.

[[- serversku -]]: MUST be a string containing SKU or edition information for the server.

[[- serverversion -]]: MUST be a string containing version information for the server.

[[- url -]]: MUST be a string containing a URL for the server.

3.1.3.3.2 Response

The response MUST be an ASCII text file that adheres to the following template.

```
<?xml version="1.0" encoding="utf-16"?>
<EnrollResponse
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <LicensorCertificateChain
         xmlns="http://microsoft.com/DRM/EnrollmentService">
         <string>[[- SLC -]]</string>
         <string>[[- EnrollmentServiceCert -]]</string>
         <string>[[- EnrollmentCACert -]]</string>
         <string>[[- CACert -]]</string>
      </LicensorCertificateChain>
</EnrollResponse>
```

[[- SLC -]]: MUST be a string containing the SLC.

[[- EnrollmentServiceCert -]]: MUST be a string containing the Enrollment Service certificate.

[[- EnrollmentCACert -]]: MUST be a string containing the Enrollment CA certificate.

[[- CACert -]]: MUST be a string containing the CA certificate.

3.1.4 Common Fault Codes

Exceptions Thrown: The RMS: Client-to-Server Protocol allows a server to notify a client of application-level faults by generating SOAP faults as specified in [SOAP1.1] section 4.4. In the SOAP fault, the <faultcode> element contains the type of exception being thrown. The <faultstring> element contains the text of the exception being thrown. The following table summarizes the default exceptions that the server can throw to the client.

Some methods may throw other exceptions to those described in this table. In each case, those exceptions are defined along with the method in the following sections.

Exception	Description
ArgumentException	.NET exception
ArgumentOutOfRangeException	.NET exception
ArgumentNullException	.NET exception
FormatException	.NET exception
UnauthorizedAccessException	The access is unauthorized.
HidXmlExpiredException	A newer version of the HID XML is required.
MismatchedIdentificationEmailAddressException	The e-mail address provided does not match the authorization identification.
TemporaryAccountCertificateException	Only one temporary account certificate can be valid at a time.
VerifyEmailAddressFailedException	The e-mail address is formatted incorrectly.
VerifyMachineCertificateChainFailedException	The machine certificate provided has a certificate chain that is not valid.
CryptoUnsupportedSymKeyException	The supplied enabling bits have an unsupported content key.
BlackBoxIsInvalidException	The client's RM lockbox has been revoked. The client computer must be reactivated to retrieve the latest RM lockbox.
DrmacIsExcludedException	The account certificate has been excluded and is not permitted to submit this request.
InvalidPersonaCertSignatureException	The account certificate the requestor supplied has been tampered with.
InvalidPersonaCertTimeException	The account certificate the requestor supplied is currently invalid.
NoRightsForRequestedPrincipalException	The publishing license contains no rights for the requested principal.
UnexpectedPersonaCertException	An unexpected error was encountered while validating the account certificate.
UntrustedPersonaCertException	The account certificate the requestor supplied was not issued by a trusted user **domain** server.
ADEntrySearchFailedException	Failed to find an entry in Active Directory.
DRMSArgumentException	An argument exception occurred. See the inner exception for details.
MalformedDataVersionException	A client request contained an invalid version number that cannot be processed.
UnsupportedDataVersionException	The data version the client requested is not supported. The server cannot process the request.

3.1.5 Authentication

The RMS system uses the user's e-mail address as a canonical identifier when specifying identities, rights, and policies. The server MUST authenticate the end user making the client request for the Certify method so that it can retrieve the user's e-mail address from a directory and include it in the RAC. The user's e-mail address MUST be included in the RAC.

The server SHOULD authenticate the end user making the FindServiceLocationsForUser method so that it can find the appropriate server for the user from the directory.

The server MAY<12> also support Microsoft Web Browser Federated Sign-On authentication, as specified in [MS-MWBF]. The client MAY follow the active client profile for Microsoft Web Browser Federated Sign-On. If Microsoft Web Browser Federated Sign-On authentication is used, the e-mail address of the authenticated user MUST be made available to the server during the Certify request.

3.1.6 Server Endpoint URLs

The server MUST expose its Web methods at specific URLs for the client to find them. The server MUST provide the following URL structure, building from a base URL. This is the minimal required structure. Case-sensitivity depends on the Web server being used to host the RMS server:

- [baseURL]/certification/Activation.asmx: Activate
- [baseURL]/certification/certification.asmx: Certify
- [baseURL]/certification/server.asmx: GetLicensorCertificate
- [baseURL]/certification/ServiceLocator.asmx: FindServiceLocationsForUser
- [baseURL]/licensing/license.asmx: AcquireLicense
- [baseURL]/licensing/publish.asmx: AcquireIssuanceLicense
- [baseURL]/licensing/publish.asmx: GetClientLicensorCert
- [baseURL]/licensing/templateDistribution.asmx: AcquireTemplateInformation
- [baseURL]/licensing/templateDistribution.asmx: AcquireTemplates
- [baseURL]/licensing/server.asmx: GetLicensorCertificate
- [baseURL]/licensing/ServiceLocator.asmx: FindServiceLocationsForUser

If the server supports Microsoft Web Browser Federated Sign-On authentication [MS-MWBF] for this protocol,<13> the following virtual directory structure MUST also exist in addition to the minimal required structure:

- [baseURL]/certificationexternal/certification.asmx: Certify
- [baseURL]/certificationexternal/server.asmx: GetLicensorCertificate
- [baseURL]/certificationexternal/ServiceLocator.asmx: FindServiceLocationsForUser
- [baseURL]/licensingexternal/license.asmx: AcquireLicense
- [baseURL]/licensingexternal/publish.asmx: AcquireIssuanceLicense
- [baseURL]/licensingexternal/publish.asmx: GetClientLicensorCert

- [baseURL]/licensingexternal/server.asmx: GetLicensorCertificate
- [baseURL]/licensingexternal/ServiceLocator.asmx: FindServiceLocationsForUser

If the server supports clients that behave as other types of servers (such as content management servers), the following virtual directory structure MUST also exist in addition to the minimal required structure:

- [baseURL]/certification/ServerCertification.asmx: Certify

If the server supports clients on mobile platforms (such as PDAs and mobile phones), the following virtual directory structure MUST also exist in addition to the minimal required structure:

- [baseURL]/certification/MobileCertification.asmx: Certify

3.1.7 Message Processing Events and Sequencing Rules

The following high-level sequence diagram illustrates the operation of the protocol.

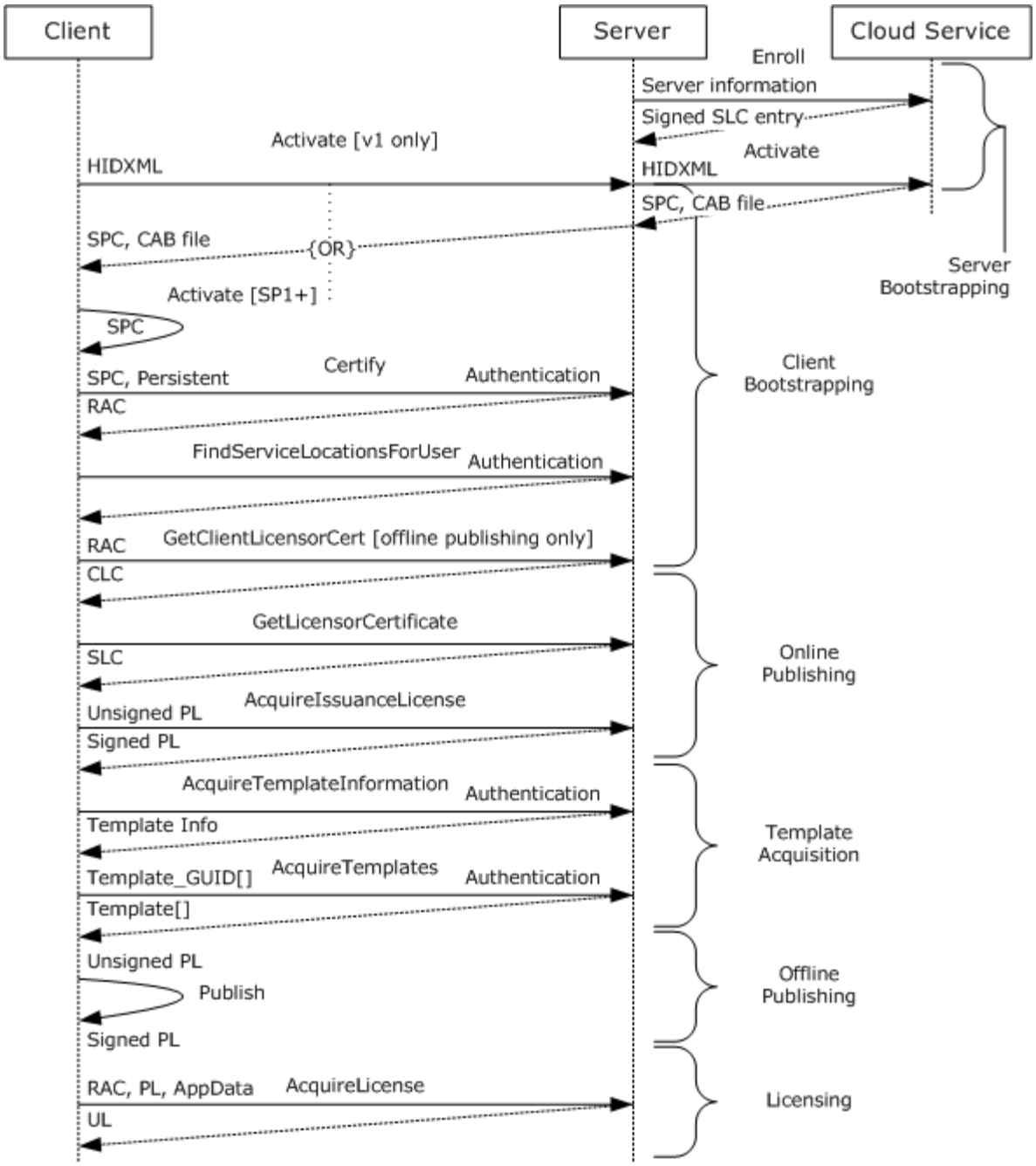

Figure 7: Protocol operation

The state data acquired from server bootstrapping previously described in section 3.1.3 MUST be retained on the server. Beyond this, no other state data is required on the server. The server MAY retain additional state data as an optimization, but it is not required. These operations are discussed in more detail in the following sections.

Note The following defined methods MUST contain a <VersionData> element in the SOAP header (as specified in [SOAP1.1]). For information on the VersionData element, see section 2.2.1.3.

3.1.7.1 Activate

During the **Activate** request, the server acts as a proxy between the version 1.0 client and the Microsoft RMS Machine Activation cloud service. The request from the client to the server and the request from the server to the cloud service are identical. Likewise, the response from the cloud service to the server and the response from the server to the client are identical.

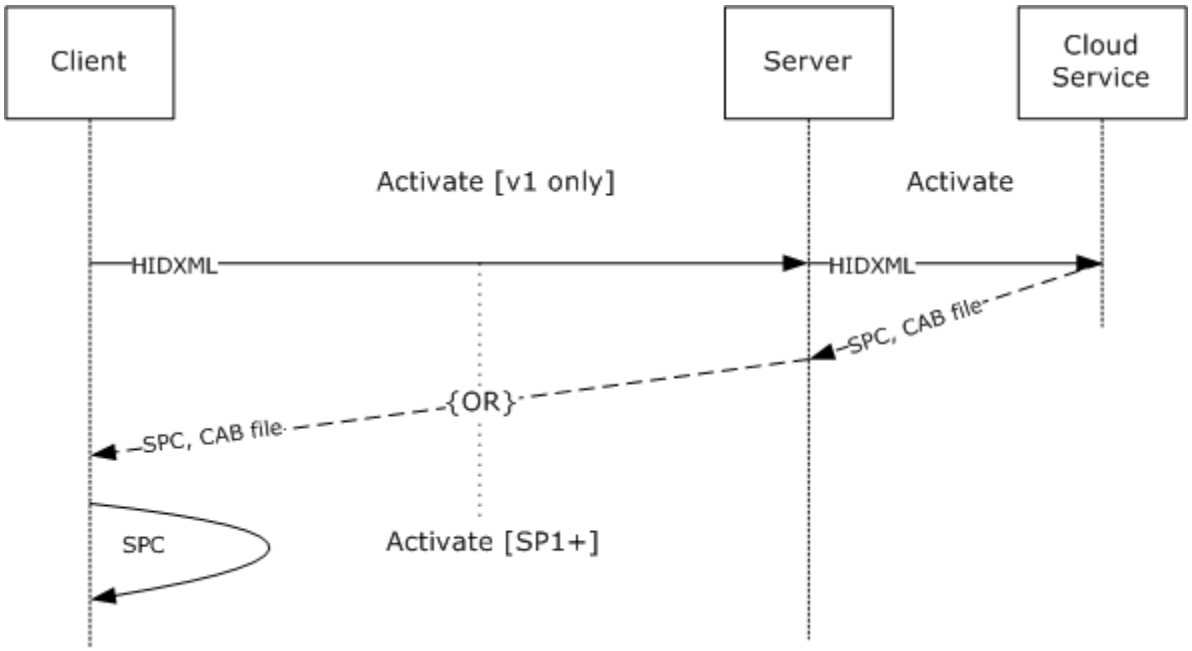

Figure 8: Activation message sequence

Request

Message Format	Description
Activate element	The Activate element, as specified in section 2.2.3.1, contains an XML structure generated by the client that contains a unique string derived from a one-way hash of hardware configuration information. This element is treated as an opaque binary large object (BLOB) by the server and forwarded to the RMS Machine Activation cloud service.

Response

Message Format	Description
ActivateResponse element	The ActivateResponse element, as defined in section 2.2.3.2, contains the SPC chain and a signature for verification of the binary data returned in a DIME attachment (as specified in [WSDLExt]). The SPC leaf-node certificate contains the public key corresponding to the private key in the security processor. This response is treated as an opaque BLOB by the server and forwarded from the RMS Machine Activation cloud service back to the client.

The **Activate** Web method response also includes binary data that the server returns verbatim as a DIME attachment to the SOAP response. This method only throws common fault codes for the RMS: Client-to-Server Protocol.

In the Activate operation, the client submits a HID hash and requests a security processor software component, signature, and SPC chain. A properly formed Activate request MUST contain a HID hash. The server treats this HID hash as an opaque BLOB and forwards it to the RMS Machine Activation cloud service.

In addition to returning an **ActivateResponse** element, the response method MAY also return a binary attachment using DIME, as specified in [WSDLExt]). The DIME attachment is treated as an opaque BLOB by the server and forwarded from the RMS Machine Activation cloud service back to the client.

The server's role in the Activate request is to act only as a proxy to the Microsoft Machine Activation cloud service. This functionality exists to enable clients that do not have connectivity to the Internet beyond the corporate environment. The Activate protocol between the server and the activation cloud service is identical to the Activate protocol between the client and the server.

Upon receiving an **Activate** request, the server SHOULD service the request. To service the request, the server MUST make an **Activate** request to the Microsoft RMS Machine Activation cloud service using the same **Activate** protocol and the same request data. The cloud service can be reached at https://activation.drm.microsoft.com/activation/activation.asmx. When the cloud service responds, the server MUST respond to the client with the same response data. The server MUST treat the request and response data as opaque BLOBs and pass the response data through to the client. A successful response includes an SPC chain, a security processor binary file containing the security processor private key, and a signature of the binary file.

After the activation step is complete, the client has a security processor with its own key pair and SPC chain.

For a successful request, the server MUST return exactly what it receives from the RMS Machine Activation cloud service. For an unsuccessful request, the server MUST return a fault code.

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData xmlns="http://microsoft.com/DRM/ActivationService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <Activate xmlns="http://microsoft.com/DRM/ActivationService">
         <requestParams>
            <ActivateParams>
               <HidXml>xml</HidXml>
            </ActivateParams>
            <ActivateParams>
               <HidXml>xml</HidXml>
            </ActivateParams>
         </requestParams>
```

```
      </Activate>
   </soap:Body>
</soap:Envelope>
```

Response Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData xmlns="http://microsoft.com/DRM/ActivationService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <ActivateResponse
         xmlns="http://microsoft.com/DRM/ActivationService">
         <ActivateResult>
            <ActivateResponse>
               <MachineCertificateChain>
                  <Certificate xsi:nil="true" />
                  <Certificate xsi:nil="true" />
               </MachineCertificateChain>
               <BinarySignature>xml</BinarySignature>
            </ActivateResponse>
         <ActivateResponse>
               <MachineCertificateChain>
                  <Certificate xsi:nil="true" />
                  <Certificate xsi:nil="true" />
               </MachineCertificateChain>
               <BinarySignature>xml</BinarySignature>
            </ActivateResponse>
         </ActivateResult>
      </ActivateResponse>
   </soap:Body>
</soap:Envelope>
```

3.1.7.2 Certify

To access protected content, the user needs a RAC that corresponds to the user's account. The RAC grants the role of a user who can access protected content. It issues an asymmetric encryption key pair and identifies the user account in the RMS system. The client uses the **Certify** request to acquire a RAC. The client MUST have a valid SPC before calling **Certify**.

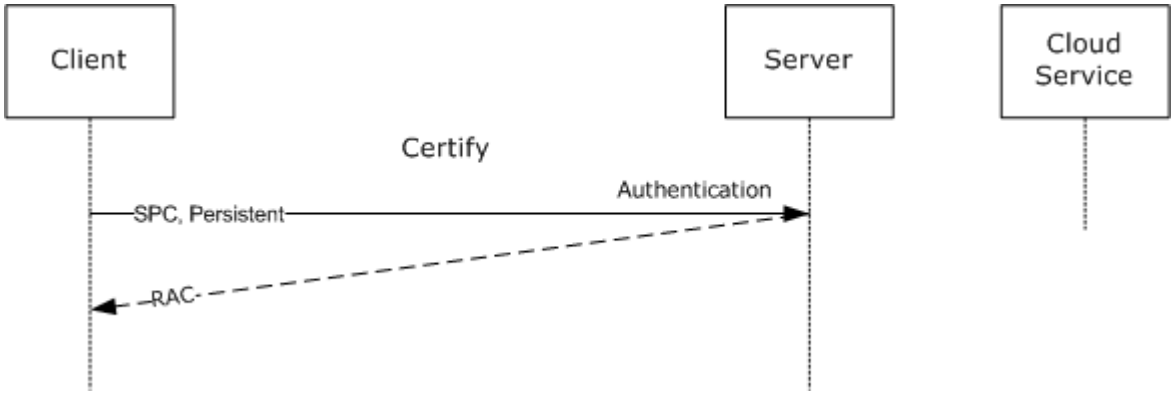

Figure 9: Certify message sequence

Request

Message Format	Description
Certify element	The Certify element, as specified in section 2.2.4.1, contains the client's SPC chain as well as a flag requesting either a persistent (long-lived) or temporary (short-lived) certificate.

Response

Message Format	Description
CertifyResponse element	The **CertifyResponse** element, as specified in section 2.2.4.2, contains the RAC chain. The RAC chain issues an encryption key pair to the user and binds the user's account to the machine through the SPC. The **CertifyResponse** element also includes a QuotaResponse structure that the client SHOULD NOT use.

Exceptions Thrown: In addition to the common RMS fault codes, the **Certify** method throws the following code:

Exception	Description
TemporaryAccountCertificateException	Only one temporary account certificate can be valid at a time.

For authentication, the request method MUST contain the user's account information acquired through an appropriate authentication scheme.<14> Because the RMS system uses a user's e-mail address as a canonical identifier, the authentication MUST provide the user's e-mail address, either directly or by allowing the server to retrieve it from a directory or other repository. The SOAP request does not encapsulate the authentication.<15>

In the **Certify** operation, the client authenticates to the server, submits an SPC chain, identifies a RAC type, and requests a RAC chain. A properly formed **Certify** request MUST contain a signed SPC chain and a flag for the RAC type.

Upon receiving a Certify request, the server SHOULD validate the signatures in the SPC chain and SHOULD verify that the trusted security processor CA key is present in the SPC chain.

If validation succeeds, the server SHOULD service the request. To service the request, the server SHOULD generate an RAC chain. To generate a RAC chain, the server MUST provide a unique

asymmetric key pair for the user. The server MAY<16> store this key pair for the user so that subsequent **Certify** requests from the same user will result in the same keys. The RAC MUST contain the user's public key in the ISSUEDPRINCIPALS element. The RAC MUST contain the user's private key, encrypted to the SPC public key, in the FEDERATIONPRINCIPALS element. The ISSUER element of the RAC MUST be copied from the ISSUEDPRINCIPALS element of the server's SLC. The SIGNATURE element of the RAC MUST be generated using the server's private key. The server's entire SLC chain MUST be appended to the RAC to form the RAC chain. For more information on the RAC chain, see section 2.3.5.

For a successful request, the server MUST return a RAC chain. For an unsuccessful request, the server MUST return a fault code. For information on Certificate formats, see section 2.3.

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Header>
         <VersionData
            xmlns="http://microsoft.com/DRM/CertificationService">
            <MinimumVersion>string</MinimumVersion>
            <MaximumVersion>string</MaximumVersion>
         </VersionData>
      </soap:Header>
      <soap:Body>
      <Certify xmlns="http://microsoft.com/DRM/CertificationService">
         <requestParams>
            <MachineCertificateChain>
               <Certificate>xml</Certificate>
               <Certificate>xml</Certificate>
            </MachineCertificateChain>
            <Persistent>boolean</Persistent>
         </requestParams>
      </Certify>
   </soap:Body>
</soap:Envelope>
```

Response Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData
       xmlns="http://microsoft.com/DRM/CertificationService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <CertifyResponse
```

```
        xmlns="http://microsoft.com/DRM/CertificationService">
          <CertifyResult>
             <CertificateChain>
                <Certificate>xml</Certificate>
                <Certificate>xml</Certificate>
             </CertificateChain>
             <Quota>
                <Verified>boolean</Verified>
                <CurrentConsumption>int</CurrentConsumption>
                <Maximum>int</Maximum>
             </Quota>
          </CertifyResult>
       </CertifyResponse>
    </soap:Body>
 </soap:Envelope>
```

3.1.7.3 FindServiceLocationsForUser

Depending on the deployment topology of the servers in the network, different servers may be used for different functions for a given user. The client SHOULD use the **FindServiceLocationsForUser** request to discover the appropriate server for various services for a given user.

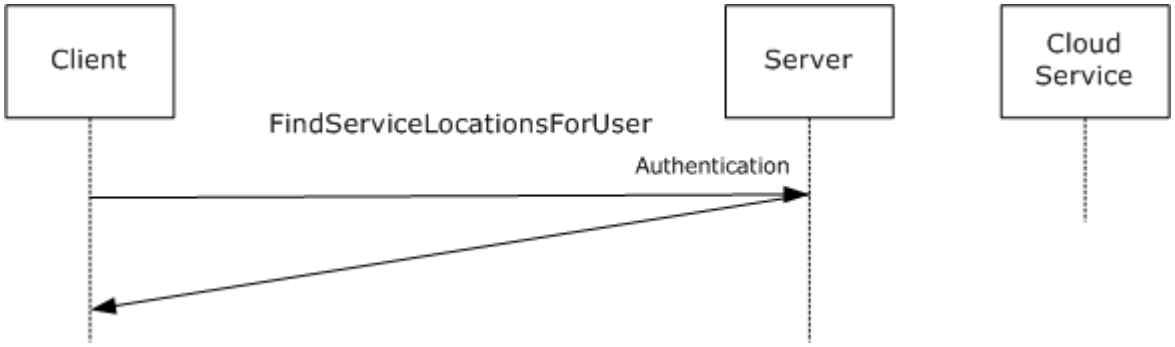

Figure 10: FindServiceLocationsForUser message sequence

Request

Message Format	Description
FindServiceLocationsForUser element	The FindServiceLocationsForUser element, as specified in section 2.2.7.1, contains a ServiceType enumeration to specify the type of service being requested.

Response

Message Format	Description
FindServiceLocationsForUserResponse element	The FindServiceLocationsForUserResponse element, as defined in section 2.2.7.2, contains the URL and ServiceType of the service that was requested.

This method throws only common fault codes for the RMS: Client-to-Server Protocol.

For authentication, the request MUST include the user's account information acquired through an appropriate authentication scheme<17> so that the server can find the appropriate service location for the user from the directory. The SOAP request does not encapsulate the authentication.

In the **FindServiceLocationsForUser** operation, the client authenticates, identifies a service type, and requests its location. A properly formed **FindServiceLocationsForUser** request MUST contain a valid ServiceType.

Upon receiving a **FindServiceLocationsForUser** request, the server SHOULD service the request. To service the request, the server SHOULD determine the locations of the appropriate server, given the user's authentication information and the requested service type, by finding the information in the directory. For a successful request, the server MUST return the appropriate service location as a URL that MAY be null for a successful request. For an unsuccessful request, the server MUST return a fault code.

The client MUST use one of the following types in the ServiceType enumeration:

- ActivationService (version 1.0 clients only)
- CertificationInternalService
- CertificationService
- LicensingService
- LicensingInternalService

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Header>
         <VersionData
            xmlns="http://microsoft.com/DRM/ServiceLocatorService">
            <MinimumVersion>string</MinimumVersion>
            <MaximumVersion>string</MaximumVersion>
         </VersionData>
      </soap:Header>
      <soap:Body>
         <FindServiceLocationsForUser
         xmlns="http://microsoft.com/DRM/ServiceLocatorService">
         <ServiceNames>
            <ServiceLocationRequest>
            <Type>EnrollmentService          or
                  LicensingService           or
                  PublishingService          or
                  CertificationService       or
                  ActivationService          or
                  PrecertificationService    or
                  ServerService              or
                  DrmRemoteDirectoryServices or
                  GroupExpansionService
            </Type>
            </ServiceLocationRequest>
```

```
            <ServiceLocationRequest>
            <Type>EnrollmentService          or
                  LicensingService           or
                  PublishingService          or
                  CertificationService       or
                  ActivationService          or
                  PrecertificationService    or
                  ServerService              or
                  DrmRemoteDirectoryServices or
                  GroupExpansionService
            </Type>
            </ServiceLocationRequest>
         </ServiceNames>
         </FindServiceLocationsForUser>
      </soap:Body>
</soap:Envelope>
```

Response Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
   <VersionData
      xmlns="http://microsoft.com/DRM/ServiceLocatorService">
      <MinimumVersion>string</MinimumVersion>
      <MaximumVersion>string</MaximumVersion>
   </VersionData>
   </soap:Header>
   <soap:Body>
      <FindServiceLocationsForUserResponse
      xmlns="http://microsoft.com/DRM/ServiceLocatorService">
      <FindServiceLocationsForUserResult>
         <ServiceLocationResponse>
            <URL>string</URL>
            <Type>EnrollmentService          or
               LicensingService           or
               PublishingService          or
               CertificationService       or
               ActivationService          or
               PrecertificationService    or
               ServerService              or
               DrmRemoteDirectoryServices or
               GroupExpansionService
            </Type>
            </ServiceLocationResponse>
            <ServiceLocationResponse>
            <URL>string</URL>
            <Type>EnrollmentService          or
               LicensingService           or
               PublishingService          or
               CertificationService       or
               ActivationService          or
```

```
                PrecertificationService    or
                ServerService              or
                DrmRemoteDirectoryServices or
                GroupExpansionService
              </Type>
              </ServiceLocationResponse>
            </FindServiceLocationsForUserResult>
          </FindServiceLocationsForUserResponse>
    </soap:Body>
  </soap:Envelope>
```

3.1.7.4 GetClientLicensorCert

To create protected content without continually contacting a server, the user needs a CLC chain that corresponds to the user's account. The CLC grants the role of a user who can create protected content on behalf of the issuing server. It issues an asymmetric signing key pair that is bound to the RAC.

The client uses the **GetClientLicensorCert** request to obtain a CLC. The client MUST have a valid RAC and SPC before calling **GetClientLicensorCert**. For more information on acquiring a RAC, see section 2.3.5. For more information on acquiring an SPC, see section 2.3.4.

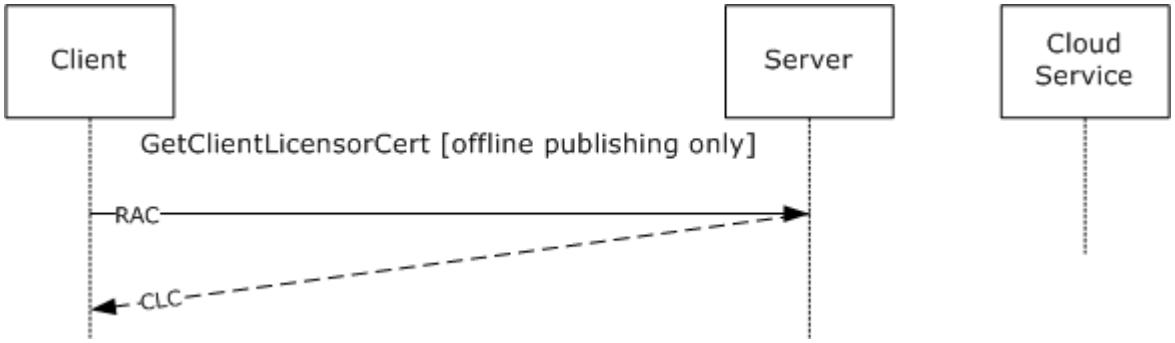

Figure 11: GetClientLicensorCert message sequence

Request

Message Format	Description
GetClientLicensorCert element	The GetClientLicensorCert element, as specified in section 2.2.6.3, contains the user's RAC chain.

Response

Message Format	Description
GetClientLicensorCertResponse element	The GetClientLicensorCertResponse element, as specified in section 2.2.6.4, contains the CLC chain. The CLC chain issues a signing key pair to the user and binds the signing keys to the user's account through the RAC.

This method throws only common fault codes for the RMS: Client-to-Server Protocol.

In the **GetClientLicensorCert** request, the client submits a RAC chain and requests a CLC chain. A properly formed **GetClientLicensorCert** request MUST contain a RAC chain.

Upon receiving a **GetClientLicensorCert** request the server SHOULD perform signature validation on the RAC chain in the request and verify that it trusts the RAC.

If validation succeeds, the server SHOULD<18> service the request by generating a CLC. To generate a CLC, the server MUST either retrieve or generate a unique asymmetric signing key pair for the user account. The server MUST encrypt the private key with the public key of the RAC so the RAC and the security processor are required to access the signing key in the CLC. The CLC MUST contain the public key and the encrypted private key. The ISSUER element of the CLC MUST contain the public key of the server. The body of the CLC MUST be signed by the server, and the signature MUST be included in the SIGNATURE element of the CLC. The server MUST append its SLC chain to the CLC to complete the CLC chain.

For a successful request, the server MUST return a CLC chain. For an unsuccessful request, the server MUST return a fault code. For information on certificate formats, see section 2.3.

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData xmlns="http://microsoft.com/DRM/PublishingService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <GetClientLicensorCert
         xmlns="http://microsoft.com/DRM/PublishingService">
         <RequestParams>
            <GetClientLicensorCertParams>
               <PersonaCerts>
                  <Certificate xsi:nil="true" />
                  <Certificate xsi:nil="true" />
               </PersonaCerts>
            </GetClientLicensorCertParams>
            <GetClientLicensorCertParams>
               <PersonaCerts>
                  <Certificate xsi:nil="true" />
                  <Certificate xsi:nil="true" />
               </PersonaCerts>
            </GetClientLicensorCertParams>
         </RequestParams>
      </GetClientLicensorCert>
   </soap:Body>
</soap:Envelope>
```

Response Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
       <VersionData xmlns="http://microsoft.com/DRM/PublishingService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
       </VersionData>
    </soap:Header>
    <soap:Body>
       <GetClientLicensorCertResponse
          xmlns="http://microsoft.com/DRM/PublishingService">
          <GetClientLicensorCertResult>
             <GetClientLicensorCertResponse>
                <CertificateChain>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </CertificateChain>
             </GetClientLicensorCertResponse>
             <GetClientLicensorCertResponse>
                   <CertificateChain>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </CertificateChain>
             </GetClientLicensorCertResponse>
          </GetClientLicensorCertResult>
       </GetClientLicensorCertResponse>
    </soap:Body>
 </soap:Envelope>
```

3.1.7.5 GetLicensorCertificate

The **GetLicensorCertificate** request is used to acquire the SLC chain from a server during online publishing. The SLC is required for online publishing because the PL MUST encrypt the usage policy and content key with the server's public key, and the SLC contains the server's public key.

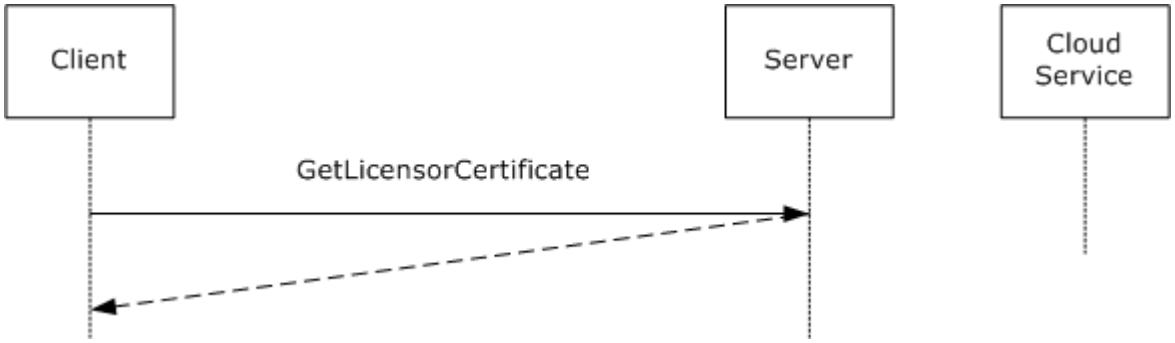

Figure 12: GetLicensorCertificate sequence

Request

Message Format	Description
GetLicensorCertificate element	The GetLicensorCertificate element, as specified in section 2.2.6.3, is an empty element that presents a request for the server's SLC chain.

Response

Message Format	Description
GetLicensorCertificateResponse element	The GetLicensorCertificateResponse element, as defined in section 2.2.6.4, contains the server's SLC chain.

This method throws only common fault codes for the RMS: Client-to-Server Protocol.

In the **GetLicensorCertificate** operation, the client requests the server's SLC chain.

Upon receiving a **GetLicensorCertificate** request, the server MUST return its SLC chain for a successful request. For an unsuccessful request, the server MUST return a fault code. For information on certificate formats, see section 2.3.

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData xmlns="http://microsoft.com/DRM/ServerService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <GetLicensorCertificate
         xmlns="http://microsoft.com/DRM/ServerService" />
   </soap:Body>
</soap:Envelope>
```

Response Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData xmlns="http://microsoft.com/DRM/ServerService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <GetLicensorCertificateResponse
```

```
            xmlns="http://microsoft.com/DRM/ServerService">
            <GetLicensorCertificateResult>
               <CertificateChain>
                  <Certificate>xml</Certificate>
                  <Certificate>xml</Certificate>
               </CertificateChain>
            </GetLicensorCertificateResult>
         </GetLicensorCertificateResponse>
      </soap:Body>
   </soap:Envelope>
```

3.1.7.6 AcquireIssuanceLicense

A PL cannot be used for licensing until it has been signed by a server. The **AcquireIssuanceLicense** request is used to sign a PL during online publishing.

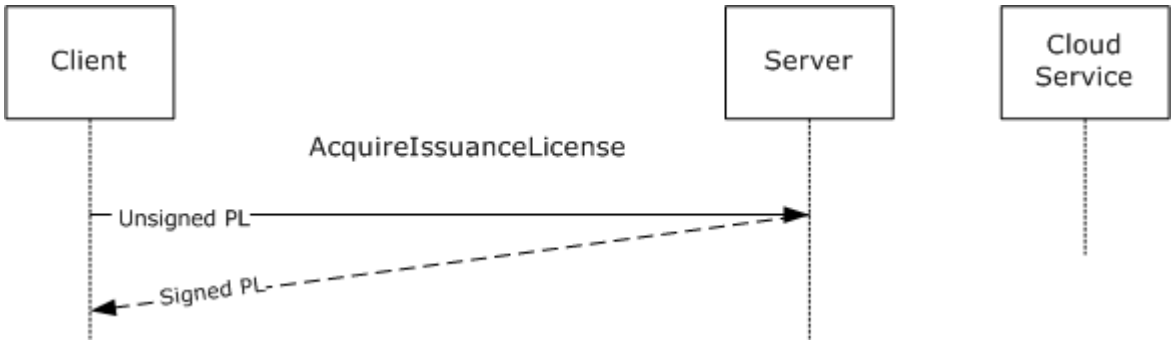

Figure 13: AcquireIssuanceLicense sequence

Request

Message Format	Description
AcquireIssuanceLicense element	The AcquireIssuanceLicense element, as specified in section 2.2.6.1, contains an unsigned PL.

Response

Message Format	Description
AcquireIssuanceLicenseResponse element	The AcquireIssuanceLicenseResponse element, as defined in section 2.2.6.2, contains a signed PL chain.

Exceptions Thrown: In addition to the common RMS fault codes, the **AcquireIssuanceLicense** method throws the following codes:

Exception	Description
InvalidTemplateSignatureException	Signature validation failed for the official template included in the PL.
OnlinePublishingDisabledException	Online publishing is not available on this

Exception	Description
	server.
UnsignedIssuanceLicenseNoMatchingIssuedPrincipalException	None of the issued principals matches this server.

In the **AcquireIssuanceLicense** operation, the client submits an unsigned PL and requests a signed PL chain. A properly formed **AcquireIssuanceLicense** request MUST contain an unsigned PL.

Upon receiving an AcquireIssuanceLicense request, the server SHOULD validate the unsigned PL for format and syntax. If any validation fails, the server SHOULD return the appropriate fault code.

If validation succeeds, the server SHOULD service the request. To service the request, the server MUST sign the body of the PL and include the signature in the SIGNATURE element of the PL. The server MUST append its leaf-node SLC to the signed PL to complete the PL chain. For information on certificate formats, see section 2.3.

For a successful request, the server MUST return a signed PL chain. For an unsuccessful request, the server MUST return a fault code.

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData xmlns="http://microsoft.com/DRM/PublishingService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <AcquireIssuanceLicense
         xmlns="http://microsoft.com/DRM/PublishingService">
         <RequestParams>
            <AcquireIssuanceLicenseParams>
               <UnsignedIssuanceLicense>xml</UnsignedIssuanceLicense>
            </AcquireIssuanceLicenseParams>
            <AcquireIssuanceLicenseParams>
               <UnsignedIssuanceLicense>xml</UnsignedIssuanceLicense>
            </AcquireIssuanceLicenseParams>
         </RequestParams>
      </AcquireIssuanceLicense>
   </soap:Body>
</soap:Envelope>
```

Response Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
       <VersionData xmlns="http://microsoft.com/DRM/PublishingService">
          <MinimumVersion>string</MinimumVersion>
          <MaximumVersion>string</MaximumVersion>
       </VersionData>
    </soap:Header>
    <soap:Body>
       <AcquireIssuanceLicenseResponse
          xmlns="http://microsoft.com/DRM/PublishingService">
          <AcquireIssuanceLicenseResult>
             <AcquireIssuanceLicenseResponse>
                <CertificateChain>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </CertificateChain>
             </AcquireIssuanceLicenseResponse>
             <AcquireIssuanceLicenseResponse>
                <CertificateChain>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </CertificateChain>
             </AcquireIssuanceLicenseResponse>
          </AcquireIssuanceLicenseResult>
       </AcquireIssuanceLicenseResponse>
    </soap:Body>
 </soap:Envelope>
```

3.1.7.7 AcquireTemplateInformation

The **AcquireTemplateInformation** request is used to acquire information about the rights policy templates available on the server. The server returns information about the available templates in the form of a list of GUIDs and hashes corresponding to the server templates.

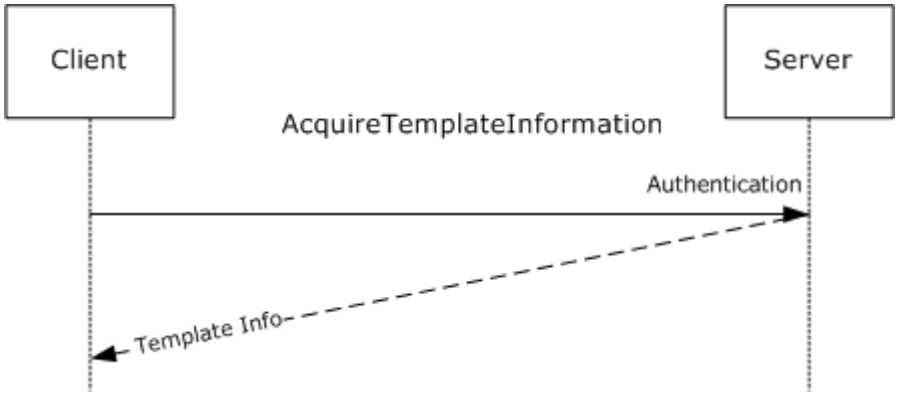

Figure 14: AcquireTemplateInformation sequence

Request

Message Format	Description
AcquireTemplateInformation	The AcquireTemplateInformation element, as defined in section 2.2.5.3, is an empty element sent to the server as part of the SOAP request.

Response

Message Format	Description
AcquireTemplateInformationResponse	The AcquireTemplateInformationResponse element, as defined in section 2.2.5.4, contains information about the rights policy templates available on the server.

In the **AcquireTemplateInformation** operation, the client requests template information from the server. The request MUST always be the same, with no specific request parameters.

Upon receiving an **AcquireTemplateInformation** request, the server SHOULD check its rights policy templates. The server SHOULD return information for its list of templates. This information MUST contain the GUID of the template and its hash value.

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Header>
    <VersionData xmlns="http://microsoft.com/DRM/TemplateDistributionService">
      <MinimumVersion>string</MinimumVersion>
      <MaximumVersion>string</MaximumVersion>
    </VersionData>
  </soap12:Header>
  <soap12:Body>
    <AcquireTemplateInformation
xmlns="http://microsoft.com/DRM/TemplateDistributionService" />
  </soap12:Body>
</soap12:Envelope>
```

Response Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Header>
    <VersionData xmlns="http://microsoft.com/DRM/TemplateDistributionService">
      <MinimumVersion>string</MinimumVersion>
      <MaximumVersion>string</MaximumVersion>
    </VersionData>
  </soap12:Header>
  <soap12:Body>
    <AcquireTemplateInformationResponse
xmlns="http://microsoft.com/DRM/TemplateDistributionService">
      <AcquireTemplateInformationResult>
        <ServerPublicKey>string</ServerPublicKey>
        <GuidHashCount>int</GuidHashCount>
        <GuidHash>
          <Guid>string</Guid>
```

```
            <Hash>string</Hash>
          </GuidHash>
          <GuidHash>
            <Guid>string</Guid>
            <Hash>string</Hash>
          </GuidHash>
        </AcquireTemplateInformationResult>
      </AcquireTemplateInformationResponse>
    </soap12:Body>
  </soap12:Envelope>
```

3.1.7.8 AcquireTemplates

The **AcquireTemplates** request is used to acquire specific rights policy templates from the server. The template can then be used to create protected content. The template describes usage policies for intended recipients when they access a particular content file protected using the template.

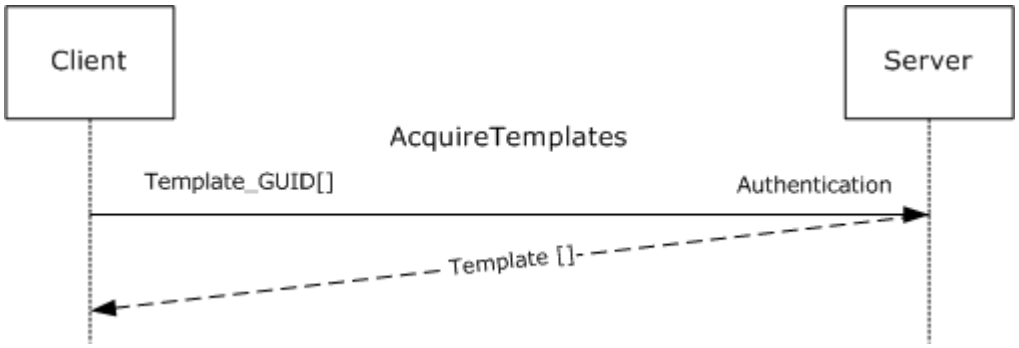

Figure 15: AcquireTemplates message sequence

Request

Message Format	Description
AcquireTemplates	The AcquireTemplates element, as defined in section 2.2.5.5, contains GUIDs of the rights policy templates that the client is requesting from the server.

Response

Message Format	Description
AcquireTemplatesResponse	The AcquireTemplatesResponse element, as defined in section 2.2.5.6, contains the rights policy templates requested by the client.

In the **AcquireTemplates** operation, the client MUST submit a list of rights policy template GUIDs and request templates corresponding to these GUIDs.

Upon receiving an **AcquireTemplates** request, the server SHOULD check whether it has the requested rights policy templates. The server SHOULD return a list of templates corresponding to the GUID list it obtained in the request. In addition to the template XML, each returned object in the list MUST include the template GUID and hash. If the server cannot find a template matching the GUID, it MUST return a null value for that template's XML field.

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Header>
    <VersionData xmlns="http://microsoft.com/DRM/TemplateDistributionService">
      <MinimumVersion>string</MinimumVersion>
      <MaximumVersion>string</MaximumVersion>
    </VersionData>
  </soap12:Header>
  <soap12:Body>
    <AcquireTemplates xmlns="http://microsoft.com/DRM/TemplateDistributionService">
      <guids>
        <string>string</string>
        <string>string</string>
      </guids>
    </AcquireTemplates>
  </soap12:Body>
</soap12:Envelope>
```

Response Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Header>
    <VersionData xmlns="http://microsoft.com/DRM/TemplateDistributionService">
      <MinimumVersion>string</MinimumVersion>
      <MaximumVersion>string</MaximumVersion>
    </VersionData>
  </soap12:Header>
  <soap12:Body>
    <AcquireTemplatesResponse
xmlns="http://microsoft.com/DRM/TemplateDistributionService">
      <AcquireTemplatesResult>
        <GuidTemplate>
          <Guid>string</Guid>
          <Hash>string</Hash>
          <Template>string</Template>
        </GuidTemplate>
        <GuidTemplate>
          <Guid>string</Guid>
          <Hash>string</Hash>
          <Template>string</Template>
        </GuidTemplate>
      </AcquireTemplatesResult>
    </AcquireTemplatesResponse>
  </soap12:Body>
</soap12:Envelope>
```

3.1.7.9 AcquireLicense

The **AcquireLicense** request is used to acquire a UL from the server. A UL is required for a user to access protected content. The UL describes what usage policies apply to the user while accessing a particular protected content file. It also contains the content key encrypted with the user's RAC public key. The UL is the authorization token that allows a user to access protected content.

Figure 16: AcquireLicense message sequence

Request

Message Format	Description
AcquireLicense element	The AcquireLicense element, as specified in section 2.2.5.1, contains the user's RAC chain and the PL chain for the content for which access is being requested.

Response

Message Format	Description
AcquireLicenseResponse element	The AcquireLicenseResponse element, as specified in section 2.2.5.2, contains the UL chain.

Exceptions Thrown: In addition to the common RMS fault codes, the **AcquireLicense** method throws the following codes.

Exception	Description
InvalidPersonaCertSignatureException	The account certificate the requestor supplied has been tampered with.
InvalidPersonaCertTimeException	The account certificate the requestor supplied is currently invalid.
UnexpectedPersonaCertException	An unexpected error was encountered while validating the account certificate.
UntrustedPersonaCertException	The account certificate the requestor supplied was not issued by a trusted user domain server.
NoRightsForRequestedPrincipalException	The PL contains no rights for the requested principal.

In the **AcquireLicense** operation, the client submits a signed PL chain, a RAC chain, and application data, and requests a UL chain. A properly formed **AcquireLicense** request MUST contain a signed PL chain, a RAC chain, and application data XML. The application data XML MAY contain a null value by way of an empty XML element.

Upon receiving an **AcquireLicense** request, the server SHOULD perform signature validation on the PL chain and ensure that it trusts the issuer of the PL. The server MUST know the private key that corresponds to the public key of the issuer of the PL in order to issue a UL. The server SHOULD

perform signature validation on the RAC chain and verify that it trusts the RAC. If any validation fails, the server SHOULD return the appropriate fault code.

If validation succeeds, the server SHOULD service the request. To service the request, the server SHOULD decrypt the usage policy and content key from the PL using its private key. The server MUST determine if the user identified by the RAC is allowed to access the content according to either the policy in the PL or another trusted policy. The server SHOULD follow any level of indirection in making this determination, such as group memberships, aliases, etc. If the user is not granted any access, the server MUST return the appropriate fault code.

If the user is granted some level of access according to the policy, the server SHOULD generate a UL to return to the client. The UL MUST describe the access that has been granted along with any conditions on that access as determined by the policy. The UL MUST contain the content key encrypted with the RAC public key. The UL SHOULD include the application data that was submitted in the request. The ISSUER element of the UL MUST contain the public key of the server. The body of the UL MUST be signed by the server, and the signature MUST be included in the SIGNATURE element of the UL. The server MUST append its SLC chain to the UL to complete the UL chain. For information on certificate formats, see section 2.3.

For a successful request, the server MUST return a UL chain. For an unsuccessful request, the server MUST return a fault code.

Request Template

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <VersionData
         xmlns="http://microsoft.com/DRM/LicensingService">
         <MinimumVersion>string</MinimumVersion>
         <MaximumVersion>string</MaximumVersion>
      </VersionData>
   </soap:Header>
   <soap:Body>
      <AcquireLicense
         xmlns="http://microsoft.com/DRM/LicensingService">
         <RequestParams>
            <AcquireLicenseParams>
               <LicenseeCerts>
                  <Certificate xsi:nil="true" />
                  <Certificate xsi:nil="true" />
               </LicenseeCerts>
               <IssuanceLicense>
                  <Certificate xsi:nil="true" />
                  <Certificate xsi:nil="true" />
               </IssuanceLicense>
               <ApplicationData>
                  xml
               </ApplicationData>
            </AcquireLicenseParams>
            <AcquireLicenseParams>
               <LicenseeCerts>
                  <Certificate xsi:nil="true" />
                  <Certificate xsi:nil="true" />
```

```
                </LicenseeCerts>
                <IssuanceLicense>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </IssuanceLicense>
                <ApplicationData>
                   xml
                </ApplicationData>
             </AcquireLicenseParams>
          </RequestParams>
       </AcquireLicense>
    </soap:Body>
 </soap:Envelope>
```

Response Template

```
 <?xml version="1.0" encoding="utf-8"?>
 <soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
       <VersionData
          xmlns="http://microsoft.com/DRM/LicensingService">
          <MinimumVersion>string</MinimumVersion>
          <MaximumVersion>string</MaximumVersion>
       </VersionData>
    </soap:Header>
    <soap:Body>
       <AcquireLicenseResponse
          xmlns="http://microsoft.com/DRM/LicensingService">
          <AcquireLicenseResult>
             <AcquireLicenseResponse>
                <CertificateChain>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </CertificateChain>
                <ReferenceCertificates>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </ReferenceCertificates>
             </AcquireLicenseResponse>
             <AcquireLicenseResponse>
                <CertificateChain>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </CertificateChain>
                <ReferenceCertificates>
                   <Certificate xsi:nil="true" />
                   <Certificate xsi:nil="true" />
                </ReferenceCertificates>
             </AcquireLicenseResponse>
          </AcquireLicenseResult>
       </AcquireLicenseResponse>
    </soap:Body>
 </soap:Envelope>
```

3.1.8 Timer Events

The RMS: Client-to-Server Protocol has no timer events.

3.1.9 Other Local Events

3.1.9.1 SLC Expiry

The SLC grants the server the right to issue certificates and licenses by way of the ISSUE RIGHT inside the WORK element of the certificate. The ISSUE RIGHT has a RANGETIME condition that specifies the range during which the SLC can be used for issuing certificates and licenses. Outside this range, the server SHOULD NOT issue certificates or licenses.

If the RANGETIME on the ISSUE RIGHT expires, the server MUST have its SLC reissued to continue functioning. To have the SLC reissued, the server SHOULD repeat the behavior specified in 3.1.3.

3.2 Client Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The organization is provided to explain how the protocol behaves. This specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this specification.

- **SPC private key**: A unique private key that is generated at activation time and issued to the machine, either by self-activation or by calling the Activate method. The private key is stored securely on the client.
- SPC chain: An XrML 1.2 certificate chain generated during activation that contains the public key corresponding to the SPC private key. The trusted security processor CA key exists in the chain.
- RAC chain: An XrML 1.2 certificate chain that issues an asymmetric encryption key pair to a user account, bound to a machine. Acquired by making a Certify request to the server.
- CLC Chain: An XrML 1.2 certificate chain that issues an asymmetric signing key pair to a user account, bound to a machine. Acquired by making a **GetClientLicensorCert** request to the server.
- PL chain: An XrML 1.2 license chain that defines usage policy for protected content and contains the content key with which that content is encrypted. Generated per protected content during publishing.
- UL chain: An XrML 1.2 license that authorizes a user to access a given protected content file and describes the usage policies that apply. Acquired per protected content by making an **AcquireLicense** request to the server.

Note that the above conceptual data can be implemented using a variety of techniques. Any data structure that stores the above conceptual data may be used in the implementation.

3.2.2 Timers

The RMS: Client-to-Server Protocol has no timer events.

3.2.3 Initialization

3.2.3.1 Client Machine Activation

The client machine MUST be certified for use with protected content. This step is called "activation" and generates an SPC key pair and certificate chain. For RMS version 1.0 clients, activation is performed by making an **Activate** request to the server. For all versions of the client beyond version 1.0, activation is performed as a client-only action without making any request to the server. The client generates its own security processor key pair and certificate chain, signed by a trusted security processor CA key that is included with the client.

3.2.3.2 Service Locations

The client MAY use any of the following discovery mechanisms to locate RMS servers:

- Active Directory.
- Existing client configuration data.
- Discovery of a server from a DISTRIBUTIONPOINT element in an existing license.

The following sections define each of the ways to discover an RMS server.

3.2.3.2.1 Locating an RMS Server by Using Existing Client Configuration Data

A client machine MAY<19> be preconfigured with stored server locations.

3.2.3.2.2 Locating an RMS Server by Using Existing Licenses or Certificates

If the client has access to an existing PL or UL, it MAY discover a server using the URL specified in the DISTRIBUTIONPOINT element in the license. If multiple URLs are specified, the client MAY try any or all of them.

To find the appropriate server for an **Activate** request, the client SHOULD make a **FindServiceLocationsForUser** request to the DISTRIBUTIONPOINT URL requesting **ServiceType** "ActivationService". This **ServiceType** is for version 1.0 clients only. All other versions of the client MUST NOT request **ServiceType** "ActivationService".

To find the appropriate server for a **Certify** request for the current user, the client SHOULD make a **FindServiceLocationsForUser** request to the DISTRIBUTIONPOINT URL requesting ServiceType "CertificationInternalService". If the response returns a URL that cannot be reached for a **Certify** request, the client SHOULD make another **FindServiceLocationsForUser** request to the DISTRIBUTIONPOINT URL requesting ServiceType "CertificationService".

To find the appropriate server for a **GetClientLicensorCert** request for the current user, the client SHOULD make a **FindServiceLocationsForUser** request to the URL requesting **ServiceType** "LicensingInternalService". If the response returns a URL that cannot be reached for a **GetClientLicensorCert** request, the client SHOULD make another **FindServiceLocationsForUser** request to the DISTRIBUTIONPOINT URL requesting **ServiceType** "LicensingInternalService".

To find the appropriate server for online publishing, the client MAY make a **FindServiceLocationsForUser** request to the URL requesting **ServiceType** "LicensingInternalService". If the response returns a URL that cannot be reached for online publishing, the client SHOULD make another **FindServiceLocationsForUser** request to the DISTRIBUTIONPOINT URL requesting **ServiceType** "LicensingInternalService".

3.2.4 Message Processing Events and Sequencing Rules

The following illustration shows a common message sequence for the client.

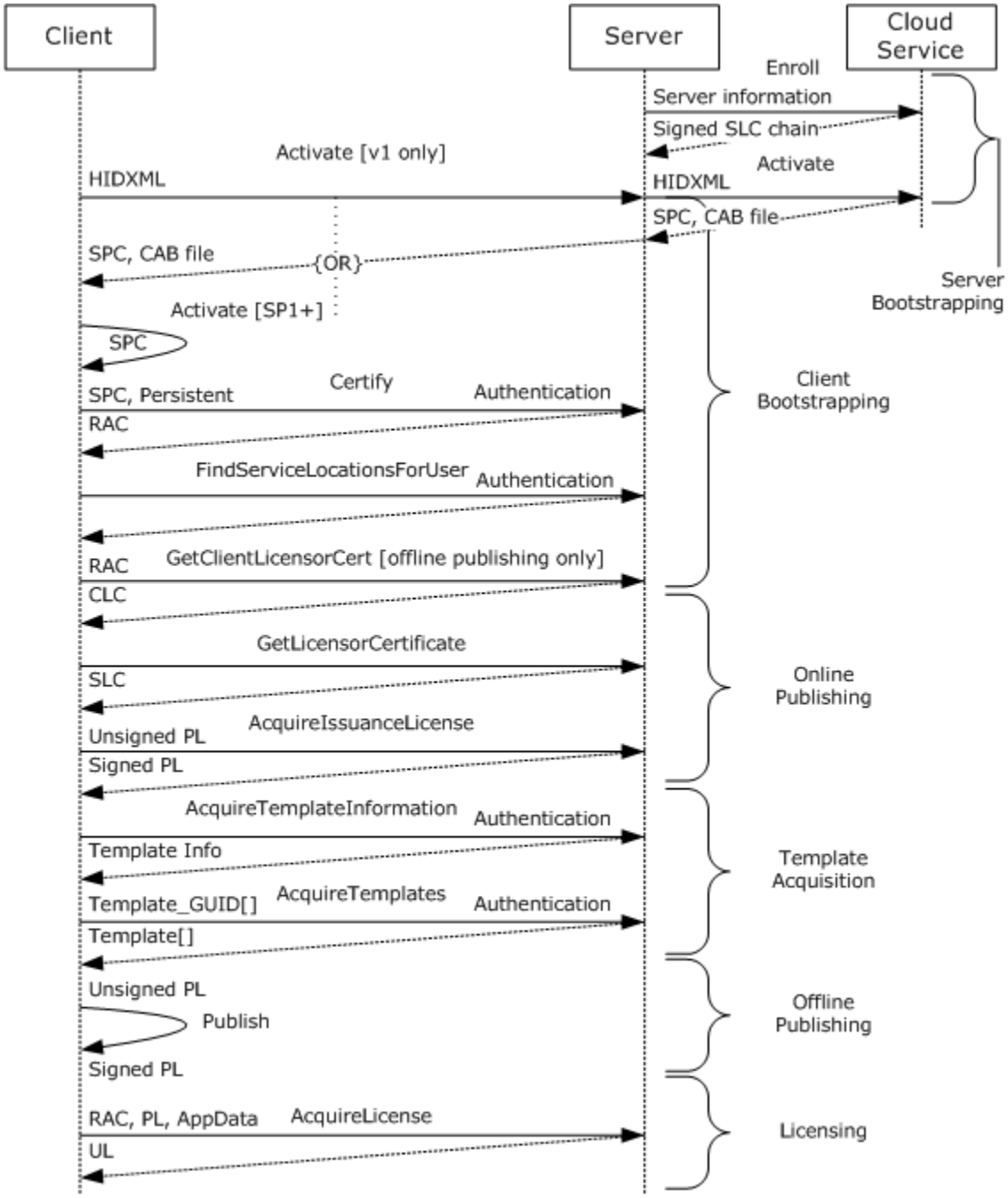

Figure 17: Common message sequence for the client

Sequencing rules for the client can be divided into four sections: client bootstrapping, online publishing, offline publishing, and licensing.

3.2.4.1 Client Bootstrapping

Client bootstrapping is required before offline publishing or licensing can take place. It is not a prerequisite for online publishing.

The client MUST activate as a first step in bootstrapping. Activation is the process of certifying a given client machine for use in the RMS system. This is accomplished by binding an encryption key pair to the machine by way of the security processor and its SPC. Version 1.0 clients MUST make an **Activate (section 3.1.7.1)** request to the server to activate. All other versions of the client, including RMS 1.0 SP1, RMS 1.0 SP2, and RMS 2.0, activate themselves without contacting a server.

The user MUST be certified to participate in the RMS system. This is accomplished by binding an encryption key pair to both the user and the client machine by way of a RAC. The user MUST have a RAC to access protected content or to publish protected content offline. The client uses the **Certify (section 3.1.7.2)** method to acquire a RAC.

To publish offline, the user MUST have a signing key pair. The CLC binds a signing key pair to a user through the RAC. A user MUST have a CLC to create protected content offline. The client uses the **FindServiceLocationsForUser (section 3.1.7.3)** method to find the licensing server for the user and the **GetClientLicensorCert (section 3.1.7.4)** method to acquire a CLC from that server.

3.2.4.2 Online Publishing

Client bootstrapping is not required for online publishing. To create a PL, the client MUST have the public key of the licensing server so it can encrypt the content key and usage policies to the correct entity. As the server's public key is stored in the SLC, the client MUST use the **GetLicensorCertificate (section 3.1.7.5)** method to acquire the server's SLC.

After the PL is constructed, it must be signed by the server before it can be used for licensing. The client MUST use the **AcquireIssuanceLicense (section 3.1.7.6)** method to have the server sign the PL.

3.2.4.3 Template Acquisition

The RMS client in Windows Vista SP1 and Windows Server 2008 MAY fetch rights policy templates from an RMS 2.0 server. The RMS client makes an AcquireTemplateInformation request to the server. The server returns information about the available templates in the form of a list of GUIDs and hashes corresponding to the server templates. The client then compares the obtained list against the list of templates from that server in its local store. The client deletes templates that are no longer present on the server. Through this process the client always keeps its templates in sync with the ones on the server.<20>

3.2.4.4 Offline Publishing

After bootstrapping is complete and the client has a valid SPC, RAC, and CLC, the client can publish protected content offline without needing to contact a server to have PLs signed. During offline publishing, the client generates a PL and signs it with the CLC private key. It also generates a UL for the owner and signs it with the CLC private key so that the owner can continue to work with the protected content without having to contact the server again. Offline publishing is the recommended method of publishing for client applications.

3.2.4.5 Licensing

To access protected content, a user MUST have a UL that binds the content key to the RAC. To acquire a UL, the client MUST use the **AcquireLicense (section 3.1.7.9)** method.

3.2.5 Timer Events

The RMS client has no timer events.

3.2.6 Other Local Events

The RMS client has no other local events.

4 Protocol Examples

4.1 Publishing Usage Policy Example

Publishing usage policy is part of the process of protecting information. Publishing usage policy is the act of expressing who may use an author's protected information, in what way, and with what conditions and durations. Published usage policy is signed by an issuer—either the server (online publishing) or the author (offline publishing). In the case of offline publishing, the server delegates the author to sign the usage policy on its behalf. The server honors this signature as a trusted delegate by issuing the author a CLC chain. The CLC represents an asymmetric key pair that is used to sign usage policy, thereby publishing it.

RMS is responsible only for issuing policy and certificates. The application (for example, the Microsoft Office System with Information Rights Management) is responsible for persisting the policy with the protected information.

The following section describes a typical scenario involving an RM-aware application and an author who is publishing usage policy for protected information:

1. Deploy client package.

   Deployment of the client package involves installing binaries on the client machine. With Windows Vista, these binaries are installed as part of the operating system. Prior to Windows Vista, a user had to download and install a separate package that deployed the client binaries.

2. Activate machine locally.

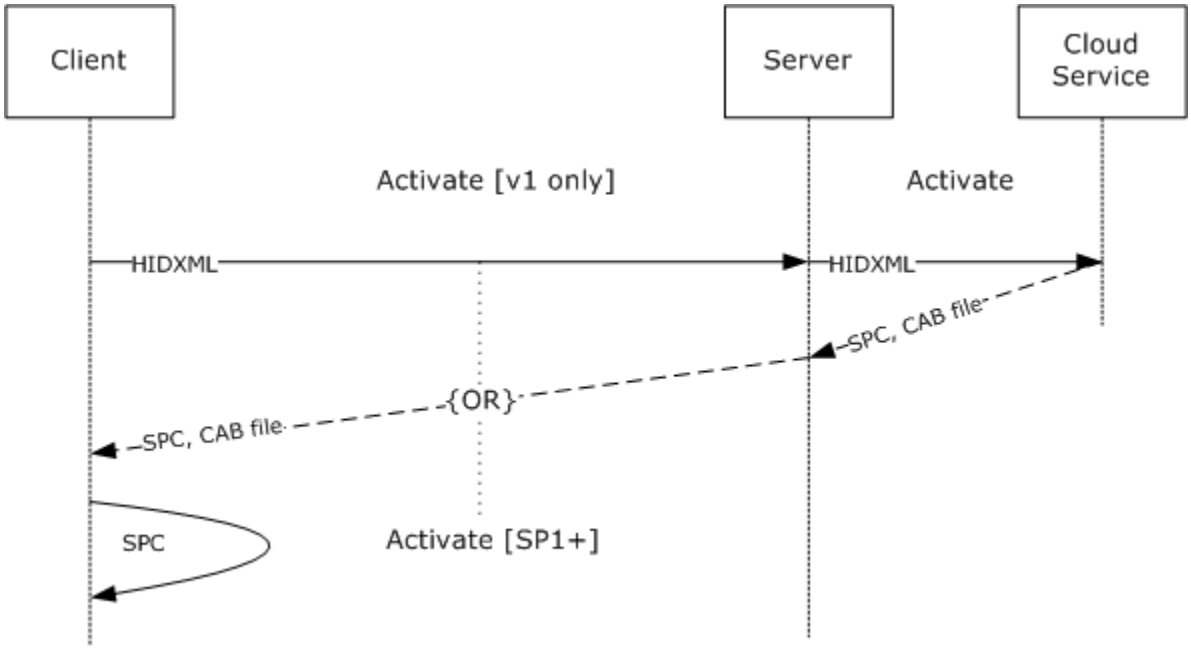

Figure 18: Local machine activation

Activation is the process by which an SPC is generated on the client machine. The SPC represents a pair of keys for the machine that are used to protect the user's keys in a subsequent step.

In RMS 1.0, the activation stage involved contacting a Web service run by Microsoft to acquire a binary and some metadata. Subsequent versions eliminated the need for this step by providing a form of self-activation that does not contact the server.

3. Call the **Certify** method.

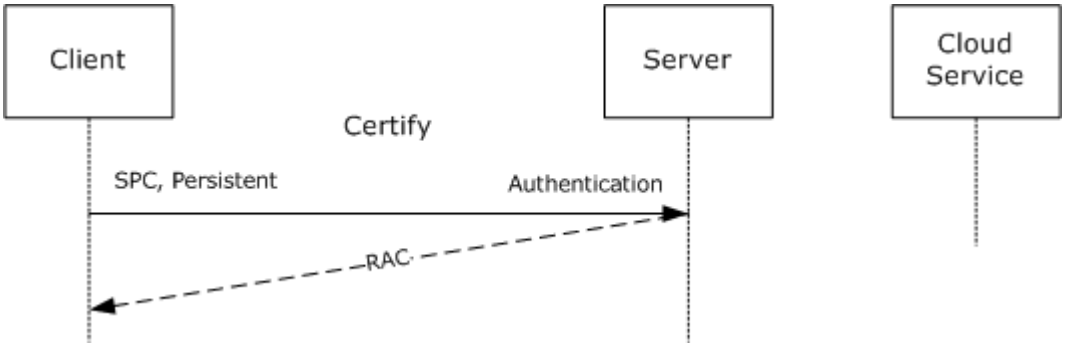

Figure 19: Certify method call

Certification is the process by which the server issues a RAC. The RAC represents a pair of keys for the user that are used to protect authorization policy and content keys in subsequent steps. The RAC keys are themselves protected by the keys represented by the SPC from step 2.

The call to the **Certify** method provides the SPC a form of authentication and a flag that indicates whether to issue a temporary, short-lived RAC or a normal, long-lived RAC. The result of a successful **Certify** call is a RAC.

4. Call the **GetClientLicensorCert** method.

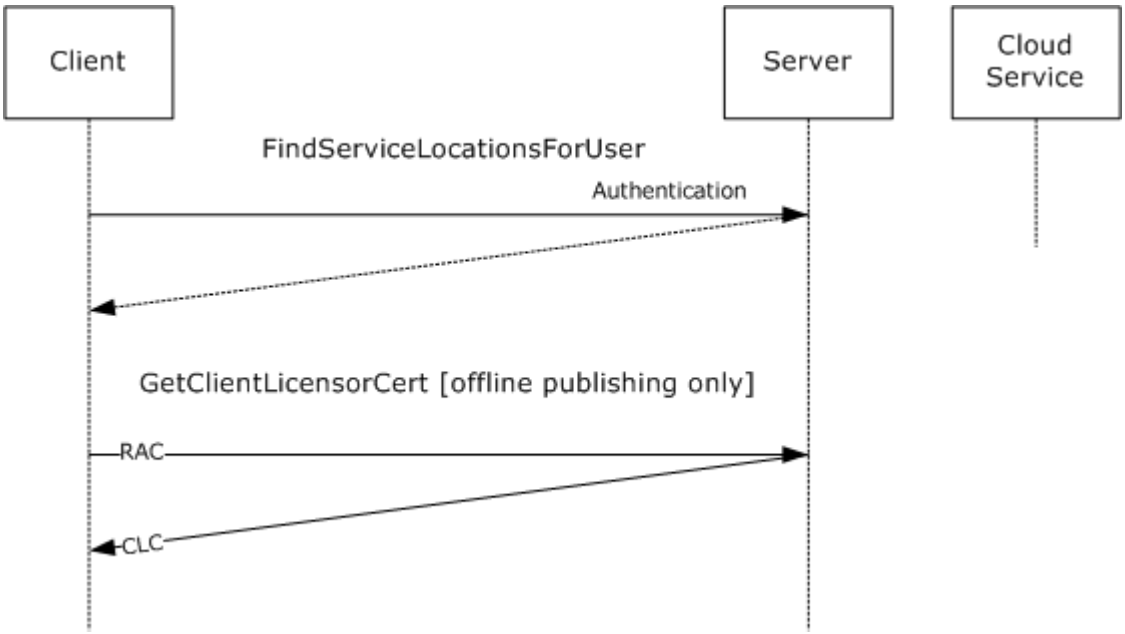

Figure 20: GetClientLicensorCert method call

To publish offline, a client must possess a CLC chain. A CLC is a form of delegation issued by the server that allows the client author to sign usage policies for protected information.

The client first calls the **FindServiceLocationsForUser** Web method, providing the authentication information, to determine at which URL the server that issues CLCs is located. Once this URL is obtained, the client calls the **GetClientLicensorCert** Web method at this URL and provides the user RAC. A successful response from the server results in a CLC being returned to the client.

5. Encrypt protected information using client APIs.

   At this point the application and the client have all certificates and keys needed to complete the publishing and protection step. The application encrypts the information using these certificates, keys, and the RMS client application programming interfaces (APIs).

6. Construct the usage policy using client APIs.

   The application uses the RM client APIs to construct the usage policy (unsigned issuance license) that expresses the set of users who may use this protected information, in what ways, and under what conditions. The usage policy can be created either directly or by using a rights policy template.

7. Sign the usage policy using client APIs and a CLC key.

   The unsigned issuance license is signed using the key represented by the CLC, producing official usage policy in the form of a signed issuance license.

8. Application persists policy with protected information.

   Finally, the application persists the signed issuance license in a location it can access along with the protected information.

4.2 Accessing Protected Information Example

Accessing protected information requires requesting an authorization policy from the RM server, and then decrypting the protected information.

1. Client package is deployed.

   Deployment of the client package involves installing binaries on the client machine. With Windows Vista, these binaries are installed as part of the operating system. Prior to Windows Vista, a user had to download and install a separate package that deployed the client binaries.

2. The machine activates locally.

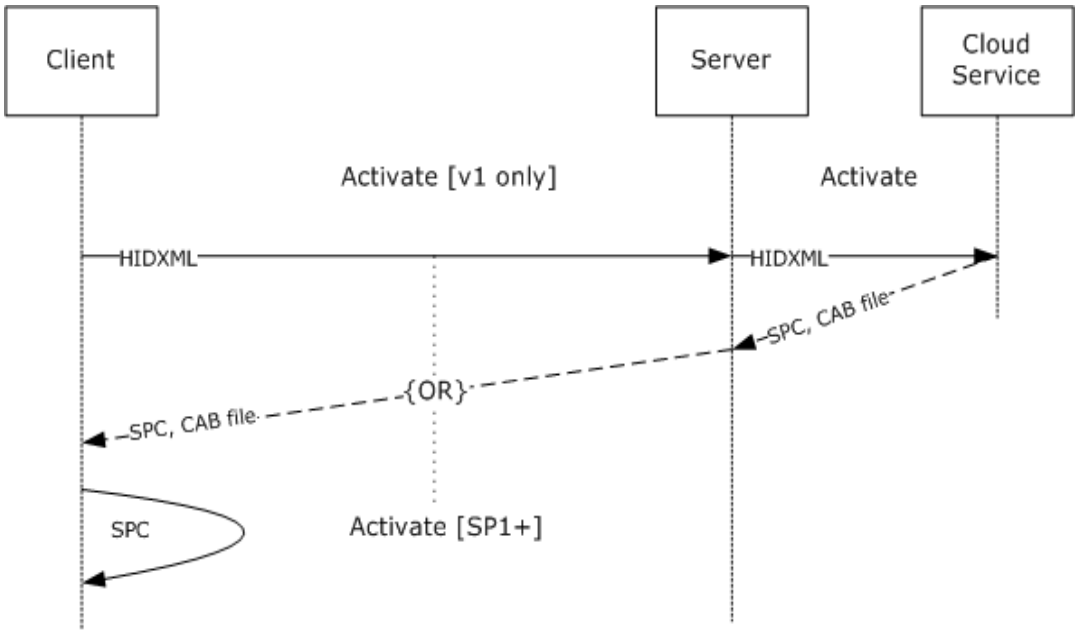

Figure 21: Local machine activation

Activation is the process by which an SPC is generated on the client machine. The SPC represents a pair of keys for the machine that are used to protect the user's keys in a subsequent step.

In RMS 1.0, the activation stage involved contacting a Web service run by Microsoft to acquire a binary and some metadata. Subsequent versions eliminated the need for this step by providing a form of self-activation that does not contact the server.

3. The **Certify** method is called.

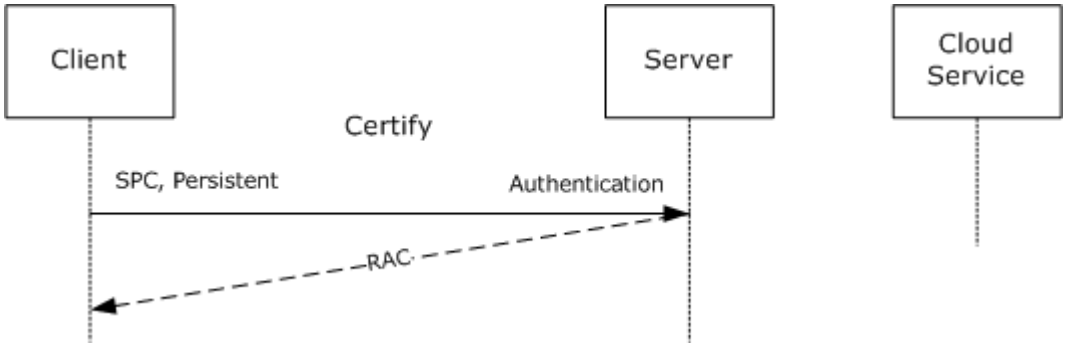

Figure 22: Certify message sequence

Certification is the process by which the server issues a RAC. The RAC represents a pair of keys for the user that are used to protect the authorization policy and content keys in subsequent steps. The RAC keys are, themselves, protected by the keys represented by the SPC from step 2.

The call to the **Certify** Web method provides the SPC a form of authentication and a flag that indicates whether to issue a temporary, short-lived RAC or a normal, long-lived RAC. The result of a successful **Certify** call is a RAC.

4. The application extracts the usage policy from the protected information.

   The application extracts or retrieves the usage policy (signed issuance license) from wherever it is stored. RMS is not responsible for storing the usage policy associated with protected information; that is the responsibility of the application.

5. The **AcquireLicense** method is called.

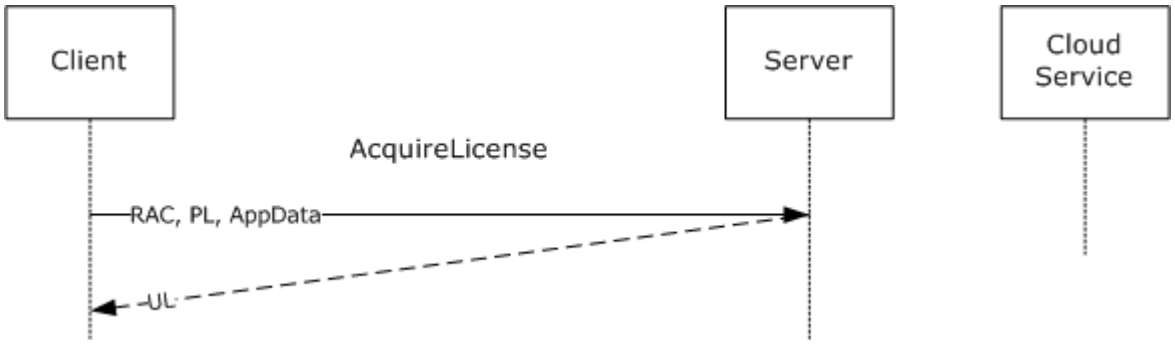

Figure 23: AcquireLicense method sequence

   The signed issuance license acquired in step 4 represents the complete usage policy issued by the author of the protected information. For an individual user to access the protected information, the server must issue an authorization policy, or use license (UL). This authorization policy expresses what an individual user can do with the protected information.

   The client calls the **AcquireLicense** Web method, providing the RAC, the signed issuance license, and passing application data that the application provided.

   The server verifies that the RAC and signed issuance license were issued from an entity or entities it trusts, and then identifies the subset of the full usage policy that applies to the specific user. It issues a UL that contains this subset of usage policy, adds the application data to the UL without modifying it, and signs the UL. The UL is then returned to the client.

6. Decryption of protected information using client APIs and authorization policy keys occurs.

   Contained within the UL issued in step 5 is the symmetric key used to protect the information. The symmetric key is encrypted to the user's RAC by the server upon issuance of the UL. The application uses the UL and the RM client APIs to decrypt the protected information and to access the information.

7. Application persists the UL with protected information.

   Finally, the application MAY persist the UL in a location it can access along with the protected information.

4.3 SOAP on DIME Response from Activate Method Example

This section shows a possible response from the **Activate** Web method, in which a DIME attachment, as specified in [DIME], is present.

DIME record 1 is as follows:

```
1 0 0 0000000000000
010 0000000101001
```

```
00000000000000000001001010101100
http://schemas.xmlsoap.org/soap/envelope/
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header>
   <VersionData xmlns="http://microsoft.com/DRM/ActivationService">
      <RequiredVersion>string</RequiredVersion>
   </VersionData>
</soap:Header>
<soap:Body>
   <ActivateResponse
      xmlns="http://microsoft.com/DRM/ActivationService">
      <ActivateResult>
         <ActivateResponse>
            <Binary = ""/>
            <BinarySignature>xml</BinarySignature>
            <MachineCertificateChain>
               <Certificate xsi:nil ="true"/>
               <Certificate xsi:nil ="true"/>
            </MachineCertificateChain>
         </ActivateResponse>
         <ActivateResponse>
            <Binary = ""/>
            <BinarySignature>xml</BinarySignature>
            <MachineCertificateChain>
               <Certificate xsi:nil ="true"/>
               <Certificate xsi:nil ="true"/>
            </MachineCertificateChain>
         </ActivateResponse>
      </ActivateResult>
   </ActivateResponse>
</soap:Body>
</soap:Envelope>
```

The DIME record is broken into three parts.

- Fixed-Length Binary Header

Element	Contents	Explanation
Record flags	1 0 0	The Message Begin (MB) flag is set.
ID length	0000000000000	No ID is set for this first record, thus the ID length is zero.
Type name format	010	The type format is a URI, expressed as 0x02.
Type length	0000000101001	The type is expressed in 41 bytes.
Data length	00000000000000000001001010101100	The data is expressed in 4,780 bytes (estimated for this example).

- Remaining Header

Element	Contents	Explanation
ID	N/A	No ID necessary for this first record.
Type	http://schemas.xmlsoap.org/soap/envelope/	The first record contains a SOAP message.

- Data

Data
The data in the first record is the SOAP response containing the machine certificate chain and a pointer to the DIME record that contains the binary data.

DIME record 2 is as follows:

```
0 0 1 0000000010000
001 0000000011000
00000000000000001111111111111111
SecureRepository
application/octet-stream
  <128KB of binary data>
```

DIME Record 2 is broken into three parts.

- Fixed-Length Binary Header

Element	Contents	Explanation
Record flags	0 0 1	The Chunked Flag (CF) is set.
ID length	0000000010000	This record is identified in 16 bytes.
Type name format	001	The type format is a MIME type, expressed as 0x01.
Type length	0000000011000	The type is expressed in 24 bytes.
Data length	00000000000000001111111111111111	The data in this record is expressed in 128 KB. The rest of the secure repository archive file is sent in the following chunked records.

- Remaining Header

Element	Contents	Explanation
ID	<GUID>	This record is identified as the beginning of the records that contain the binary data. This GUID is automatically generated.
Type	application/octet-stream	This record is purely binary, used to transmit the binary data.

- Data

Data
For the purposes of this example, the binary data is taken to be 158,974 bytes in size. This example is transmitting the binary in 128-KB chunks, so this first chunked record contains 128 KB of binary data.

DIME record 3 is as follows:

```
0 1 0 0000000000000
000 0000000000000
00000000000000000000110110011111111
  <27903 bytes of binary data>
```

Record 3 is also broken into three parts.

- Fixed-Length Binary Header

Element	Contents	Explanation
Record flags	0 1 0	The Message End (ME) flag is set and the CF is cleared, denoting this message as the end of the chunked binary and the end of the DIME response.
ID length	0000000000000	All chunked records inherit the ID of the first chunked record; thus this is zero.
Type name format	000	All chunked records inherit the type of the first chunked record; thus this is zero.
Type length	0000000011000	All chunked records inherit the type of the first chunked record; thus this is zero.
Data length	00000000000000000000110110011111111	The data in this record is expressed in 27,903 bytes as the final record in this chunked transfer.

- Remaining Header

Element	Contents	Explanation
ID		All chunked records inherit the ID of the first chunked record; thus this is empty.
Type		All chunked records inherit the type of the first chunked record; thus this is empty.

- Data

Data
For the purposes of this example, the binary data is taken to be 158,974 bytes in size. Since 128 KB were transmitted in the previous record, 27,903 bytes remain.

4.4 Template Acquisition Example

Template acquisition is a process by which client machines keep their local copy of templates in sync with the server.

The following section describes a typical scenario where the client synchronizes its local templates with those on the server.

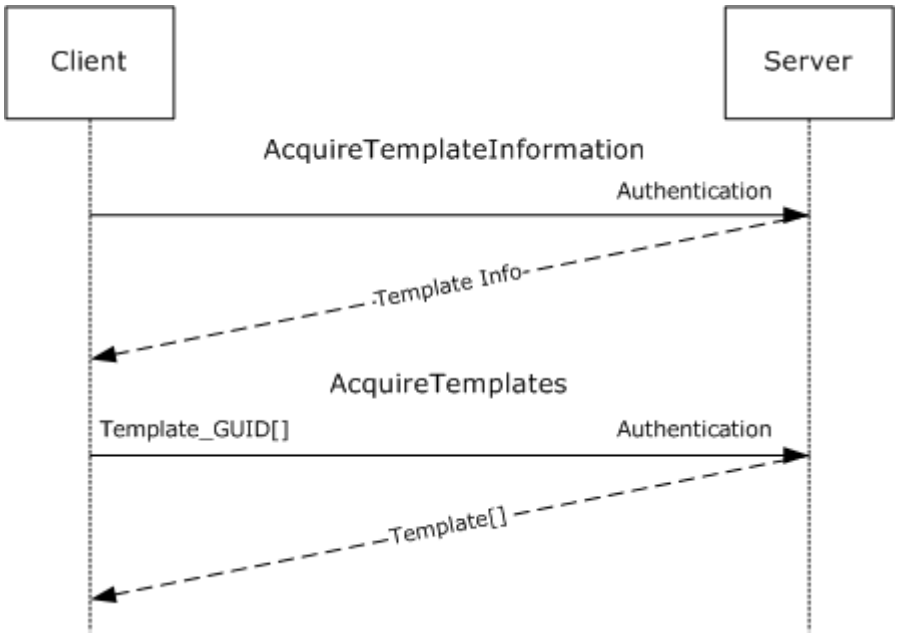

Figure 24: State diagram for client template synchronization

AcquireTemplateInformation: The client initially makes an **AcquireTemplateInformation** request to the server. The server returns information about the available templates in the form of a list of GUIDs and hashes for all the server templates. The client then compares the obtained list against the list of templates from that server in its local store. The client deletes templates that are no longer present on the server.
AcquireTemplates: For the templates that are either not present in the local store or that have been updated on the server, the client makes an **AcquireTemplates** request. This request sends a list of GUIDs to the server indicating the templates that the client is requesting. The server then returns the requested templates to the client. On obtaining these templates, the client puts them in the local store.

5 Security

5.1 Security Considerations for Implementers

Certificate signatures are generated by computing a SHA1 hash of the contents of the body element (including start and end tags) of a certificate. The hash is then signed using an asymmetric key pair. The keys, digest, and encryption algorithm used all conform to RSA PKCS#1 version 1.5, as specified in [PKCS1].

Single-DES is deprecated and SHOULD NOT be used. AES is preferred.

5.2 Index of Security Parameters

Security parameter	Section
Transport authentication	2.1
Encryption algorithms	2.3.1.13

6 Appendix A: Full WSDL Definitions

For ease of implementation, this section provides the full WSDL. The syntax uses the XrML syntax extensions, as specified in [WSDL].

6.1 Activation Service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:tns="http://microsoft.com/DRM/ActivationService"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    targetNamespace="http://microsoft.com/DRM/ActivationService"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
      targetNamespace="http://microsoft.com/DRM/ActivationService">
      <s:element name="Activate">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
                name="requestParams"
                type="tns:ArrayOfActivateParams" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfActivateParams">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded"
            name="ActivateParams" nillable="true"
            type="tns:ActivateParams" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="ActivateParams">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="HidXml">
            <s:complexType mixed="true">
              <s:sequence>
                <s:any />
              </s:sequence>
            </s:complexType>
          </s:element>
        </s:sequence>
      </s:complexType>
      <s:element name="ActivateResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
                name="ActivateResult"
                type="tns:ArrayOfActivateResponse" />
          </s:sequence>
        </s:complexType>
```

```
    </s:element>
    <s:complexType name="ArrayOfActivateResponse">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="unbounded"
          name="ActivateResponse"
          type="tns:ActivateResponse" />
      </s:sequence>
    </s:complexType>
    <s:complexType name="ActivateResponse">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="MachineCertificateChain"
          type="tns:ArrayOfXmlNode" />
        <s:element minOccurs="0" maxOccurs="1"
              name="BinarySignature">
          <s:complexType mixed="true">
            <s:sequence>
              <s:any />
            </s:sequence>
          </s:complexType>
        </s:element>
      </s:sequence>
    </s:complexType>
    <s:complexType name="ArrayOfXmlNode">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="unbounded"
              name="Certificate"
              nillable="true">
          <s:complexType mixed="true">
            <s:sequence>
              <s:any />
            </s:sequence>
          </s:complexType>
        </s:element>
      </s:sequence>
    </s:complexType>
    <s:element name="VersionData" type="tns:VersionData" />
    <s:complexType name="VersionData">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="MinimumVersion" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
          name="MaximumVersion" type="s:string" />
      </s:sequence>
      <s:anyAttribute />
    </s:complexType>
  </s:schema>
</wsdl:types>
<wsdl:message name="ActivateSoapIn">
  <wsdl:part name="parameters" element="tns:Activate" />
</wsdl:message>
<wsdl:message name="ActivateSoapOut">
  <wsdl:part name="parameters" element="tns:ActivateResponse" />
</wsdl:message>
<wsdl:message name="ActivateVersionData">
  <wsdl:part name="VersionData" element="tns:VersionData" />
</wsdl:message>
<wsdl:portType name="ActivationProxyWebServiceSoap">
```

```
    <wsdl:operation name="Activate">
      <wsdl:input message="tns:ActivateSoapIn" />
      <wsdl:output message="tns:ActivateSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ActivationProxyWebServiceSoap"
    type="tns:ActivationProxyWebServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="Activate">
      <soap:operation
soapAction="http://microsoft.com/DRM/ActivationService/Activate"
       style="document" />
      <wsdl:input>
        <soap:body use="literal" />
        <soap:header message="tns:ActivateVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
        <soap:header message="tns:ActivateVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ActivationProxyWebServiceSoap12"
        type="tns:ActivationProxyWebServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Activate">
      <soap12:operation
      soapAction="http://microsoft.com/DRM/ActivationService/Activate"
        style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
        <soap12:header message="tns:ActivateVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
        <soap12:header message="tns:ActivateVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ActivationProxyWebService">
    <wsdl:port name="ActivationProxyWebServiceSoap"
        binding="tns:ActivationProxyWebServiceSoap">
      <soap:address
   location="http://rmsx86e716/_wmcs/certification/activation.asmx" />
    </wsdl:port>
    <wsdl:port name="ActivationProxyWebServiceSoap12"
          binding="tns:ActivationProxyWebServiceSoap12">
      <soap12:address
   location="http://rmsx86e716/_wmcs/certification/activation.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

6.2 Certification Service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
   xmlns:tns="http://microsoft.com/DRM/CertificationService"
   xmlns:s="http://www.w3.org/2001/XMLSchema"
   xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   targetNamespace="http://microsoft.com/DRM/CertificationService"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
     targetNamespace="http://microsoft.com/DRM/CertificationService">
      <s:element name="Certify">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
               name="requestParams"
               type="tns:CertifyParams" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="CertifyParams">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
             name="MachineCertificateChain"
             type="tns:ArrayOfXmlNode" />
          <s:element minOccurs="1" maxOccurs="1"
             name="Persistent"
             type="s:boolean" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="ArrayOfXmlNode">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded"
             name="Certificate" nillable="true">
            <s:complexType mixed="true">
              <s:sequence>
                <s:any />
              </s:sequence>
            </s:complexType>
          </s:element>
        </s:sequence>
      </s:complexType>
      <s:element name="CertifyResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
               name="CertifyResult"
               type="tns:CertifyResponse" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="CertifyResponse">
        <s:sequence>
```

```
        <s:element minOccurs="0" maxOccurs="1"
          name="CertificateChain" type="tns:ArrayOfXmlNode" />
        <s:element minOccurs="0" maxOccurs="1"
          name="Quota" type="tns:QuotaResponse" />
      </s:sequence>
    </s:complexType>
    <s:complexType name="QuotaResponse">
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1"
          name="Verified" type="s:boolean" />
        <s:element minOccurs="1" maxOccurs="1"
          name="CurrentConsumption" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1"
          name="Maximum" type="s:int" />
      </s:sequence>
    </s:complexType>
    <s:element name="VersionData" type="tns:VersionData" />
    <s:complexType name="VersionData">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="MinimumVersion" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
          name="MaximumVersion" type="s:string" />
      </s:sequence>
      <s:anyAttribute />
    </s:complexType>
  </s:schema>
</wsdl:types>
<wsdl:message name="CertifySoapIn">
  <wsdl:part name="parameters" element="tns:Certify" />
</wsdl:message>
<wsdl:message name="CertifySoapOut">
  <wsdl:part name="parameters" element="tns:CertifyResponse" />
</wsdl:message>
<wsdl:message name="CertifyVersionData">
  <wsdl:part name="VersionData" element="tns:VersionData" />
</wsdl:message>
<wsdl:portType name="CertificationWebServiceSoap">
  <wsdl:operation name="Certify">
    <wsdl:input message="tns:CertifySoapIn" />
    <wsdl:output message="tns:CertifySoapOut" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CertificationWebServiceSoap"
    type="tns:CertificationWebServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="Certify">
    <soap:operation
 soapAction="http://microsoft.com/DRM/CertificationService/Certify"
 style="document" />
    <wsdl:input>
      <soap:body use="literal" />
      <soap:header message="tns:CertifyVersionData"
       part="VersionData" use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
      <soap:header message="tns:CertifyVersionData"
```

```
          part="VersionData" use="literal" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="CertificationWebServiceSoap12"
        type="tns:CertificationWebServiceSoap">
      <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="Certify">
        <soap12:operation
     soapAction="http://microsoft.com/DRM/CertificationService/Certify"
     style="document" />
        <wsdl:input>
          <soap12:body use="literal" />
          <soap12:header message="tns:CertifyVersionData"
              part="VersionData" use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap12:body use="literal" />
          <soap12:header message="tns:CertifyVersionData"
              part="VersionData" use="literal" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="CertificationWebService">
      <wsdl:port name="CertificationWebServiceSoap"
           binding="tns:CertificationWebServiceSoap">
        <soap:address
location="http://rmsx86e716/_wmcs/certification/certification.asmx"/>
      </wsdl:port>
      <wsdl:port name="CertificationWebServiceSoap12"
binding="tns:CertificationWebServiceSoap12">
        <soap12:address
location="http://rmsx86e716/_wmcs/certification/certification.asmx"/>
      </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

6.3 Licensing Service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
   xmlns:tns="http://microsoft.com/DRM/LicensingService"
   xmlns:s="http://www.w3.org/2001/XMLSchema"
   xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   targetNamespace="http://microsoft.com/DRM/LicensingService"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
      targetNamespace="http://microsoft.com/DRM/LicensingService">
      <s:element name="AcquireLicense">
        <s:complexType>
          <s:sequence>
```

```
      <s:element minOccurs="0" maxOccurs="1"
         name="RequestParams"
         type="tns:ArrayOfAcquireLicenseParams" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfAcquireLicenseParams">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
      name="AcquireLicenseParams"
      nillable="true"
      type="tns:AcquireLicenseParams" />
  </s:sequence>
</s:complexType>
<s:complexType name="AcquireLicenseParams">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1"
      name="LicenseeCerts" type="tns:ArrayOfXmlNode" />
    <s:element minOccurs="0" maxOccurs="1"
      name="IssuanceLicense" type="tns:ArrayOfXmlNode" />
    <s:element minOccurs="0" maxOccurs="1"
      name="ApplicationData">
      <s:complexType mixed="true">
        <s:sequence>
          <s:any />
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:sequence>
</s:complexType>
<s:complexType name="ArrayOfXmlNode">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
         name="Certificate" nillable="true">
      <s:complexType mixed="true">
        <s:sequence>
          <s:any />
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:sequence>
</s:complexType>
<s:element name="AcquireLicenseResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
         name="AcquireLicenseResult"
         type="tns:ArrayOfAcquireLicenseResponse" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfAcquireLicenseResponse">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
      name="AcquireLicenseResponse"
      nillable="true"
      type="tns:AcquireLicenseResponse" />
  </s:sequence>
```

```
    </s:complexType>
    <s:complexType name="AcquireLicenseResponse">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="CertificateChain"
          type="tns:ArrayOfXmlNode" />
        <s:element minOccurs="0" maxOccurs="1"
          name="ReferenceCertificates"
          type="tns:ArrayOfXmlNode" />
      </s:sequence>
    </s:complexType>
    <s:element name="VersionData" type="tns:VersionData" />
    <s:complexType name="VersionData">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="MinimumVersion" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
          name="MaximumVersion" type="s:string" />
      </s:sequence>
      <s:anyAttribute />
    </s:complexType>
    <s:complexType name="ArrayOfString">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="unbounded"
          name="string" nillable="true" type="s:string" />
      </s:sequence>
    </s:complexType>
          <s:complexType name="ArrayOfAcquirePreLicenseResponse">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="unbounded"
          name="AcquirePreLicenseResponse"
          nillable="true"
          type="tns:AcquirePreLicenseResponse" />
      </s:sequence>
    </s:complexType>
  </s:schema>
</wsdl:types>
<wsdl:message name="AcquireLicenseSoapIn">
  <wsdl:part name="parameters" element="tns:AcquireLicense" />
</wsdl:message>
<wsdl:message name="AcquireLicenseSoapOut">
  <wsdl:part name="parameters"
    element="tns:AcquireLicenseResponse" />
</wsdl:message>
<wsdl:message name="AcquireLicenseVersionData">
  <wsdl:part name="VersionData" element="tns:VersionData" />
</wsdl:message>
<wsdl:portType name="LicenseSoap">
  <wsdl:operation name="AcquireLicense">
    <wsdl:input message="tns:AcquireLicenseSoapIn" />
    <wsdl:output message="tns:AcquireLicenseSoapOut" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="LicenseSoap" type="tns:LicenseSoap">
  <soap:binding
    transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="AcquireLicense">
    <soap:operation soapAction=
```

```
"http://microsoft.com/DRM/LicensingService/AcquireLicense"
         style="document" />
      <wsdl:input>
        <soap:body use="literal" />
        <soap:header message="tns:AcquireLicenseVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
        <soap:header message="tns:AcquireLicenseVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="LicenseSoap12" type="tns:LicenseSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="AcquireLicense">
      <soap12:operation soapAction=
"http://microsoft.com/DRM/LicensingService/AcquireLicense"
         style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
        <soap12:header message="tns:AcquireLicenseVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
        <soap12:header message="tns:AcquireLicenseVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="AcquirePreLicense">
      <soap12:operation soapAction=
"http://microsoft.com/DRM/LicensingService/AcquirePreLicense"
         style="document" />
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="License">
    <wsdl:port name="LicenseSoap" binding="tns:LicenseSoap">
      <soap:address
         location="http://rmsx86e716/_wmcs/licensing/license.asmx" />
    </wsdl:port>
    <wsdl:port name="LicenseSoap12" binding="tns:LicenseSoap12">
      <soap12:address
         location="http://rmsx86e716/_wmcs/licensing/license.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

6.4 Template Distribution Service

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://microsoft.com/DRM/TemplateDistributionService"
```

```
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://microsoft.com/DRM/TemplateDistributionService"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://microsoft.com/DRM/TemplateDistributionService">
      <s:element name="AcquireTemplateInformation">
        <s:complexType />
      </s:element>
      <s:element name="AcquireTemplateInformationResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
name="AcquireTemplateInformationResult" type="tns:TemplateInformation" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="TemplateInformation">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="ServerPublicKey" type="s:string"
/>
          <s:element minOccurs="1" maxOccurs="1" name="GuidHashCount" type="s:int" />
          <s:element minOccurs="0" maxOccurs="unbounded" name="GuidHash"
type="tns:GuidHash" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="GuidHash">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="Guid" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Hash" type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:element name="VersionData" type="tns:VersionData" />
      <s:complexType name="VersionData">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="MinimumVersion" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="MaximumVersion" type="s:string" />
        </s:sequence>
        <s:anyAttribute />
      </s:complexType>
      <s:element name="AcquireTemplates">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="guids" type="tns:ArrayOfString"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfString">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded" name="string" nillable="true"
type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:element name="AcquireTemplatesResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="AcquireTemplatesResult"
type="tns:ArrayOfGuidTemplate" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfGuidTemplate">
        <s:sequence>
```

```
          <s:element minOccurs="0" maxOccurs="unbounded" name="GuidTemplate"
nillable="true" type="tns:GuidTemplate" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="GuidTemplate">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="Guid" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Hash" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Template" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="AcquireTemplateInformationSoapIn">
    <wsdl:part name="parameters" element="tns:AcquireTemplateInformation" />
  </wsdl:message>
  <wsdl:message name="AcquireTemplateInformationSoapOut">
    <wsdl:part name="parameters" element="tns:AcquireTemplateInformationResponse" />
  </wsdl:message>
  <wsdl:message name="AcquireTemplateInformationVersionData">
    <wsdl:part name="VersionData" element="tns:VersionData" />
  </wsdl:message>
  <wsdl:message name="AcquireTemplatesSoapIn">
    <wsdl:part name="parameters" element="tns:AcquireTemplates" />
  </wsdl:message>
  <wsdl:message name="AcquireTemplatesSoapOut">
    <wsdl:part name="parameters" element="tns:AcquireTemplatesResponse" />
  </wsdl:message>
  <wsdl:message name="AcquireTemplatesVersionData">
    <wsdl:part name="VersionData" element="tns:VersionData" />
  </wsdl:message>
  <wsdl:portType name="TemplateDistributionWebServiceSoap">
    <wsdl:operation name="AcquireTemplateInformation">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Return template
information (GUID + hash)</wsdl:documentation>
      <wsdl:input message="tns:AcquireTemplateInformationSoapIn" />
      <wsdl:output message="tns:AcquireTemplateInformationSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="AcquireTemplates">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Return
templates</wsdl:documentation>
      <wsdl:input message="tns:AcquireTemplatesSoapIn" />
      <wsdl:output message="tns:AcquireTemplatesSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="TemplateDistributionWebServiceSoap"
type="tns:TemplateDistributionWebServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="AcquireTemplateInformation">
      <soap:operation
soapAction="http://microsoft.com/DRM/TemplateDistributionService/AcquireTemplateInformati
on" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
        <soap:header message="tns:AcquireTemplateInformationVersionData"
part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
        <soap:header message="tns:AcquireTemplateInformationVersionData"
part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="AcquireTemplates">
```

```
      <soap:operation
soapAction="http://microsoft.com/DRM/TemplateDistributionService/AcquireTemplates"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
        <soap:header message="tns:AcquireTemplatesVersionData" part="VersionData"
use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
        <soap:header message="tns:AcquireTemplatesVersionData" part="VersionData"
use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="TemplateDistributionWebServiceSoap12"
type="tns:TemplateDistributionWebServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="AcquireTemplateInformation">
      <soap12:operation
soapAction="http://microsoft.com/DRM/TemplateDistributionService/AcquireTemplateInformati
on" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
        <soap12:header message="tns:AcquireTemplateInformationVersionData"
part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
        <soap12:header message="tns:AcquireTemplateInformationVersionData"
part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="AcquireTemplates">
      <soap12:operation
soapAction="http://microsoft.com/DRM/TemplateDistributionService/AcquireTemplates"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
        <soap12:header message="tns:AcquireTemplatesVersionData" part="VersionData"
use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
        <soap12:header message="tns:AcquireTemplatesVersionData" part="VersionData"
use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="TemplateDistributionWebService">
    <wsdl:port name="TemplateDistributionWebServiceSoap"
binding="tns:TemplateDistributionWebServiceSoap">
      <soap:address
location="http://rmsx86e721/ wmcs/licensing/templateDistribution.asmx" />
    </wsdl:port>
    <wsdl:port name="TemplateDistributionWebServiceSoap12"
binding="tns:TemplateDistributionWebServiceSoap12">
      <soap12:address
location="http://rmsx86e721/ wmcs/licensing/templateDistribution.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

6.5 Publishing Service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
   xmlns:tns="http://microsoft.com/DRM/PublishingService"
   xmlns:s="http://www.w3.org/2001/XMLSchema"
   xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   targetNamespace="http://microsoft.com/DRM/PublishingService"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
      targetNamespace="http://microsoft.com/DRM/PublishingService">
      <s:element name="AcquireIssuanceLicense">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
               name="RequestParams"
               type="tns:ArrayOfAcquireIssuanceLicenseParams" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfAcquireIssuanceLicenseParams">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded"
            name="AcquireIssuanceLicenseParams"
            nillable="true"
            type="tns:AcquireIssuanceLicenseParams" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="AcquireIssuanceLicenseParams">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
               name="UnsignedIssuanceLicense">
            <s:complexType mixed="true">
              <s:sequence>
                <s:any />
              </s:sequence>
            </s:complexType>
          </s:element>
        </s:sequence>
      </s:complexType>
      <s:element name="AcquireIssuanceLicenseResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
               name="AcquireIssuanceLicenseResult"
               type="tns:ArrayOfAcquireIssuanceLicenseResponse" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfAcquireIssuanceLicenseResponse">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded"
```

```
      name="AcquireIssuanceLicenseResponse"
      nillable="true"
      type="tns:AcquireIssuanceLicenseResponse" />
  </s:sequence>
</s:complexType>
<s:complexType name="AcquireIssuanceLicenseResponse">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1"
      name="CertificateChain"
      type="tns:ArrayOfXmlNode" />
  </s:sequence>
</s:complexType>
<s:complexType name="ArrayOfXmlNode">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
         name="Certificate"
         nillable="true">
      <s:complexType mixed="true">
        <s:sequence>
          <s:any />
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:sequence>
</s:complexType>
<s:element name="VersionData" type="tns:VersionData" />
<s:complexType name="VersionData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1"
      name="MinimumVersion"
      type="s:string" />
    <s:element minOccurs="0" maxOccurs="1"
      name="MaximumVersion"
      type="s:string" />
  </s:sequence>
  <s:anyAttribute />
</s:complexType>
<s:element name="GetClientLicensorCert">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
         name="RequestParams"
         type="tns:ArrayOfGetClientLicensorCertParams" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfGetClientLicensorCertParams">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
      name="GetClientLicensorCertParams"
      nillable="true"
      type="tns:GetClientLicensorCertParams" />
  </s:sequence>
</s:complexType>
<s:complexType name="GetClientLicensorCertParams">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="PersonaCerts"
         type="tns:ArrayOfXmlNode" />
```

```
      </s:sequence>
    </s:complexType>
    <s:element name="GetClientLicensorCertResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
             name="GetClientLicensorCertResult"
             type="tns:ArrayOfGetClientLicensorCertResponse" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:complexType name="ArrayOfGetClientLicensorCertResponse">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="unbounded"
          name="GetClientLicensorCertResponse"
          nillable="true"
          type="tns:GetClientLicensorCertResponse" />
      </s:sequence>
    </s:complexType>
    <s:complexType name="GetClientLicensorCertResponse">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="CertificateChain"
          type="tns:ArrayOfXmlNode" />
      </s:sequence>
    </s:complexType>
  </s:schema>
</wsdl:types>
<wsdl:message name="AcquireIssuanceLicenseSoapIn">
  <wsdl:part name="parameters"
    element="tns:AcquireIssuanceLicense" />
</wsdl:message>
<wsdl:message name="AcquireIssuanceLicenseSoapOut">
  <wsdl:part name="parameters"
    element="tns:AcquireIssuanceLicenseResponse" />
</wsdl:message>
<wsdl:message name="AcquireIssuanceLicenseVersionData">
  <wsdl:part name="VersionData"
    element="tns:VersionData" />
</wsdl:message>
<wsdl:message name="GetClientLicensorCertSoapIn">
  <wsdl:part name="parameters"
       element="tns:GetClientLicensorCert" />
</wsdl:message>
<wsdl:message name="GetClientLicensorCertSoapOut">
  <wsdl:part name="parameters"
       element="tns:GetClientLicensorCertResponse" />
</wsdl:message>
<wsdl:message name="GetClientLicensorCertVersionData">
  <wsdl:part name="VersionData" element="tns:VersionData" />
</wsdl:message>
<wsdl:portType name="PublishSoap">
  <wsdl:operation name="AcquireIssuanceLicense">
    <wsdl:input message="tns:AcquireIssuanceLicenseSoapIn" />
    <wsdl:output message="tns:AcquireIssuanceLicenseSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="GetClientLicensorCert">
    <wsdl:input message="tns:GetClientLicensorCertSoapIn" />
```

```
      <wsdl:output message="tns:GetClientLicensorCertSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="PublishSoap" type="tns:PublishSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="AcquireIssuanceLicense">
      <soap:operation soapAction=
"http://microsoft.com/DRM/PublishingService/AcquireIssuanceLicense"
         style="document" />
      <wsdl:input>
        <soap:body use="literal" />
        <soap:header message="tns:AcquireIssuanceLicenseVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
        <soap:header message="tns:AcquireIssuanceLicenseVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetClientLicensorCert">
      <soap:operation soapAction=
"http://microsoft.com/DRM/PublishingService/GetClientLicensorCert"
         style="document" />
      <wsdl:input>
        <soap:body use="literal" />
        <soap:header message="tns:GetClientLicensorCertVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
        <soap:header message="tns:GetClientLicensorCertVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="PublishSoap12" type="tns:PublishSoap">
    <soap12:binding
      transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="AcquireIssuanceLicense">
      <soap12:operation soapAction=
"http://microsoft.com/DRM/PublishingService/AcquireIssuanceLicense"
         style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
        <soap12:header
            message="tns:AcquireIssuanceLicenseVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
        <soap12:header
            message="tns:AcquireIssuanceLicenseVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetClientLicensorCert">
      <soap12:operation soapAction=
```

```
"http://microsoft.com/DRM/PublishingService/GetClientLicensorCert"
         style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
        <soap12:header message="tns:GetClientLicensorCertVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
        <soap12:header message="tns:GetClientLicensorCertVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Publish">
    <wsdl:port name="PublishSoap" binding="tns:PublishSoap">
      <soap:address
         location="http://rmsx86e716/_wmcs/licensing/publish.asmx" />
    </wsdl:port>
    <wsdl:port name="PublishSoap12" binding="tns:PublishSoap12">
      <soap12:address
         location="http://rmsx86e716/_wmcs/licensing/publish.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

6.6 Server Service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
   xmlns:tns="http://microsoft.com/DRM/ServerService"
   xmlns:s="http://www.w3.org/2001/XMLSchema"
   xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   targetNamespace="http://microsoft.com/DRM/ServerService"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
         targetNamespace="http://microsoft.com/DRM/ServerService">
      <s:element name="GetLicensorCertificate">
        <s:complexType />
      </s:element>
      <s:element name="GetLicensorCertificateResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
               name="GetLicensorCertificateResult"
               type="tns:LicensorCertChain" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="LicensorCertChain">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
```

```
      name="CertificateChain"
      type="tns:ArrayOfXmlNode" />
  </s:sequence>
</s:complexType>
<s:complexType name="ArrayOfXmlNode">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
         name="Certificate"
         nillable="true">
      <s:complexType mixed="true">
        <s:sequence>
          <s:any />
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:sequence>
</s:complexType>
<s:element name="VersionData" type="tns:VersionData" />
<s:complexType name="VersionData">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1"
      name="MinimumVersion" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1"
      name="MaximumVersion" type="s:string" />
  </s:sequence>
  <s:anyAttribute />
</s:complexType>
<s:element name="FindServiceLocations">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
         name="ServiceNames"
         type="tns:ArrayOfServiceLocationRequest" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfServiceLocationRequest">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
      name="ServiceLocationRequest" nillable="true"
      type="tns:ServiceLocationRequest" />
  </s:sequence>
</s:complexType>
<s:complexType name="ServiceLocationRequest">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Type"
      type="tns:ServiceType" />
  </s:sequence>
</s:complexType>
<s:simpleType name="ServiceType">
  <s:restriction base="s:string">
    <s:enumeration value="EnrollmentService" />
    <s:enumeration value="LicensingService" />
    <s:enumeration value="PublishingService" />
    <s:enumeration value="CertificationService" />
    <s:enumeration value="ActivationService" />
    <s:enumeration value="PrecertificationService" />
    <s:enumeration value="ServerService" />
```

```
    <s:enumeration value="DrmRemoteDirectoryServices" />
    <s:enumeration value="GroupExpansionService" />
  </s:restriction>
</s:simpleType>
<s:element name="FindServiceLocationsResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
         name="FindServiceLocationsResult"
         type="tns:ArrayOfServiceLocationResponse" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfServiceLocationResponse">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
      name="ServiceLocationResponse" nillable="true"
      type="tns:ServiceLocationResponse" />
  </s:sequence>
</s:complexType>
<s:complexType name="ServiceLocationResponse">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="URL"
      type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="Type"
      type="tns:ServiceType" />
  </s:sequence>
</s:complexType>
<s:element name="GetServerInfo">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="requests"
         type="tns:ArrayOfServerInfoRequest" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfServerInfoRequest">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="unbounded"
      name="ServerInfoRequest" nillable="true"
      type="tns:ServerInfoRequest" />
  </s:sequence>
</s:complexType>
<s:complexType name="ServerInfoRequest">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Type"
      type="tns:ServerInfoType" />
    <s:element minOccurs="0" maxOccurs="1"
      name="AdditionalInfo"
      type="s:string" />
  </s:sequence>
</s:complexType>
<s:simpleType name="ServerInfoType">
  <s:restriction base="s:string">
    <s:enumeration value="VersionInfo" />
    <s:enumeration value="ServerFeatureInfo" />
    <s:enumeration value="ServerLicensorCertificate" />
    <s:enumeration value="ServiceLocations" />
```

```
      </s:restriction>
    </s:simpleType>
    <s:element name="GetServerInfoResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
                name="GetServerInfoResult">
            <s:complexType mixed="true">
              <s:sequence>
                <s:any />
              </s:sequence>
            </s:complexType>
          </s:element>
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:schema>
</wsdl:types>
<wsdl:message name="GetLicensorCertificateSoapIn">
  <wsdl:part name="parameters"
    element="tns:GetLicensorCertificate" />
</wsdl:message>
<wsdl:message name="GetLicensorCertificateSoapOut">
  <wsdl:part name="parameters"
    element="tns:GetLicensorCertificateResponse" />
</wsdl:message>
<wsdl:message name="GetLicensorCertificateVersionData">
  <wsdl:part name="VersionData" element="tns:VersionData" />
</wsdl:message>
<wsdl:message name="FindServiceLocationsSoapIn">
  <wsdl:part name="parameters"
    element="tns:FindServiceLocations" />
</wsdl:message>
<wsdl:message name="FindServiceLocationsSoapOut">
  <wsdl:part name="parameters"
    element="tns:FindServiceLocationsResponse" />
</wsdl:message>
<wsdl:message name="FindServiceLocationsVersionData">
  <wsdl:part name="VersionData" element="tns:VersionData" />
</wsdl:message>
<wsdl:message name="GetServerInfoSoapIn">
  <wsdl:part name="parameters" element="tns:GetServerInfo" />
</wsdl:message>
<wsdl:message name="GetServerInfoSoapOut">
  <wsdl:part name="parameters"
    element="tns:GetServerInfoResponse" />
</wsdl:message>
<wsdl:portType name="ServerSoap">
  <wsdl:operation name="GetLicensorCertificate">
    <wsdl:input message="tns:GetLicensorCertificateSoapIn" />
    <wsdl:output message="tns:GetLicensorCertificateSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="FindServiceLocations">
    <wsdl:input message="tns:FindServiceLocationsSoapIn" />
    <wsdl:output message="tns:FindServiceLocationsSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="GetServerInfo">
    <wsdl:input message="tns:GetServerInfoSoapIn" />
```

```
      <wsdl:output message="tns:GetServerInfoSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ServerSoap" type="tns:ServerSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetLicensorCertificate">
      <soap:operation soapAction=
"http://microsoft.com/DRM/ServerService/GetLicensorCertificate"
        style="document" />
      <wsdl:input>
        <soap:body use="literal" />
        <soap:header message="tns:GetLicensorCertificateVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
        <soap:header message="tns:GetLicensorCertificateVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="FindServiceLocations">
      <soap:operation soapAction=
"http://microsoft.com/DRM/ServerService/FindServiceLocations"
        style="document" />
      <wsdl:input>
        <soap:body use="literal" />
        <soap:header message="tns:FindServiceLocationsVersionData"
            part="VersionData" use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
        <soap:header message="tns:FindServiceLocationsVersionData"
            part="VersionData" use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetServerInfo">
      <soap:operation
 soapAction="http://microsoft.com/DRM/ServerService/GetServerInfo"
        style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ServerSoap12" type="tns:ServerSoap">
    <soap12:binding
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="GetLicensorCertificate">
      <soap12:operation soapAction=
"http://microsoft.com/DRM/ServerService/GetLicensorCertificate"
        style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
        <soap12:header
            message="tns:GetLicensorCertificateVersionData"
```

```
            part="VersionData" use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap12:body use="literal" />
          <soap12:header
              message="tns:GetLicensorCertificateVersionData"
              part="VersionData" use="literal" />
        </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="FindServiceLocations">
        <soap12:operation soapAction=
"http://microsoft.com/DRM/ServerService/FindServiceLocations"
           style="document" />
        <wsdl:input>
          <soap12:body use="literal" />
          <soap12:header
           message="tns:FindServiceLocationsVersionData"
           part="VersionData" use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap12:body use="literal" />
          <soap12:header message="tns:FindServiceLocationsVersionData"
              part="VersionData" use="literal" />
        </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetServerInfo">
        <soap12:operation
soapAction="http://microsoft.com/DRM/ServerService/GetServerInfo"
           style="document" />
        <wsdl:input>
          <soap12:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap12:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="Server">
      <wsdl:port name="ServerSoap" binding="tns:ServerSoap">
        <soap:address
           location="http://rmsx86e716/_wmcs/licensing/server.asmx" />
      </wsdl:port>
      <wsdl:port name="ServerSoap12" binding="tns:ServerSoap12">
        <soap12:address
           location="http://rmsx86e716/_wmcs/licensing/server.asmx" />
      </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

6.7 Enrollment Cloud Service WSDL

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions
   xmlns:s1="http://microsoft.com/wsdl/types/"
   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:s="http://www.w3.org/2001/XMLSchema"
```

```
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://microsoft.com/DRM/EnrollmentService"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://microsoft.com/DRM/EnrollmentService"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
  <s:schema elementFormDefault="qualified"
        targetNamespace="http://microsoft.com/DRM/EnrollmentService">
    <s:import namespace="http://microsoft.com/wsdl/types/" />
  <s:element name="Enroll">
  <s:complexType>
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="oInput"
        type="tns:EnrollParameters" />
    </s:sequence>
    </s:complexType>
    </s:element>
  <s:complexType name="EnrollParameters">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1"
        name="AuthorizationInformation"
        type="tns:X509Information" />
    <s:element minOccurs="1" maxOccurs="1"
        name="RevocationInformation"
        type="tns:EnrolleeRevocationInformation" />
    <s:element minOccurs="1" maxOccurs="1"
        name="CertificatePublicKey"
        type="tns:EnrolleeCertificatePublicKey" />
    <s:element minOccurs="1" maxOccurs="1"
        name="EnrolleeInformation"
        type="tns:EnrolleeServerInformation" />
    </s:sequence>
    </s:complexType>
  <s:complexType name="X509Information">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1"
        name="SignedDataBase64Encoded"
        type="s:string" />
    </s:sequence>
    </s:complexType>
  <s:complexType name="EnrolleeRevocationInformation">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1"
        name="RevocationType"
        type="tns:RevocationTypeEnum" />
    <s:element minOccurs="0" maxOccurs="1"
        name="aRevocationAuthorities"
        type="tns:ArrayOfRevocationAuthorityInformation" />
    </s:sequence>
    </s:complexType>
  <s:simpleType name="RevocationTypeEnum">
  <s:restriction base="s:string">
    <s:enumeration value="NonRevocable" />
    <s:enumeration value="StandardRevocation" />
    <s:enumeration value="CustomRevocation" />
    </s:restriction>
    </s:simpleType>
```

```
<s:complexType name="ArrayOfRevocationAuthorityInformation">
<s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded"
      name="RevocationAuthorityInformation"
      type="tns:RevocationAuthorityInformation" />
  </s:sequence>
  </s:complexType>
<s:complexType name="RevocationAuthorityInformation">
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1"
      name="aRevocationAuthorityPublicKey"
      type="s:base64Binary" />
  </s:sequence>
  </s:complexType>
<s:complexType name="EnrolleeCertificatePublicKey">
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1"
      name="aPublicKeyBytes"
      type="s:base64Binary" />
  <s:element minOccurs="1" maxOccurs="1"
      name="Guid"
      type="s1:guid" />
  </s:sequence>
  </s:complexType>
<s:complexType name="EnrolleeServerInformation">
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="SKU"
      type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Version"
      type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Name"
      type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="URL"
      type="s:string" />
  </s:sequence>
  </s:complexType>
<s:element name="EnrollResponse">
<s:complexType>
<s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="EnrollResult"
      type="tns:EnrollResponse" />
  </s:sequence>
  </s:complexType>
  </s:element>
<s:complexType name="EnrollResponse">
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1"
      name="LicensorCertificateChain"
      type="tns:ArrayOfString" />
  </s:sequence>
  </s:complexType>
<s:complexType name="ArrayOfString">
<s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded"
      name="string"
      nillable="true"
      type="s:string" />
  </s:sequence>
```

```
  </s:complexType>
  <s:element name="VersionData" type="tns:VersionData" />
<s:complexType name="VersionData">
<s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="MinimumVersion"
      type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="MaximumVersion"
      type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:schema>
<s:schema elementFormDefault="qualified"
   targetNamespace="http://microsoft.com/wsdl/types/">
<s:simpleType name="guid">
<s:restriction base="s:string">
  <s:pattern value=
"[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-
[0-9a-fA-F]{12}"/>
  </s:restriction>
  </s:simpleType>
  </s:schema>
  </wsdl:types>
<wsdl:message name="EnrollSoapIn">
  <wsdl:part name="parameters" element="tns:Enroll" />
  </wsdl:message>
<wsdl:message name="EnrollSoapOut">
  <wsdl:part name="parameters" element="tns:EnrollResponse" />
  </wsdl:message>
<wsdl:message name="EnrollVersionData">
  <wsdl:part name="VersionData" element="tns:VersionData" />
  </wsdl:message>
<wsdl:portType name="EnrollServiceSoap">
<wsdl:operation name="Enroll">
  <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">
      Enrollment Entry Point
   </documentation>
  <wsdl:input message="tns:EnrollSoapIn" />
  <wsdl:output message="tns:EnrollSoapOut" />
  </wsdl:operation>
  </wsdl:portType>
<wsdl:binding name="EnrollServiceSoap" type="tns:EnrollServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
      style="document" />
<wsdl:operation name="Enroll">
  <soap:operation
      soapAction="http://microsoft.com/DRM/EnrollmentService/Enroll"
      style="document" />
<wsdl:input>
  <soap:body use="literal" />
  <soap:header message="tns:EnrollVersionData" part="VersionData"
      use="literal" />
  </wsdl:input>
<wsdl:output>
  <soap:body use="literal" />
  <soap:header message="tns:EnrollVersionData" part="VersionData"
      use="literal" />
  </wsdl:output>
  </wsdl:operation>
```

```
  </wsdl:binding>
<wsdl:service name="EnrollService">
  <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">
      A Web service used to enroll the first DRM server in an
      enterprise
   </documentation>
<wsdl:port name="EnrollServiceSoap" binding="tns:EnrollServiceSoap">
  <soap:address location=
"https://activation.drm.microsoft.com/enrollment/enrollservice.asmx"
     />
  </wsdl:port>
  </wsdl:service>
```

7 Appendix B: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1.7: The only capability currently versioned in the Windows implementation is the ability to batch multiple requests into a single client/server round-trip. Currently, the server's minimum version is 1.0.0.0 and maximum version is 1.1.0.0. Batching capabilities are available with version 1.1.0.0 or higher.

<2> Section 2.1: Protocol messages are transported using the HTTP or HTTPS protocol between client and server. Standard ports for these protocols SHOULD be used. The Windows Rights Management client and Rights Management server MUST use the same transport protocol. The RMS: Client-to-Server Protocol does not directly manipulate network layers below the transport layer.

<3> Section 2.1: RMS supports HTTPS for securing its ports, although Secure Sockets Layer (SSL) is not configured by default when RMS is installed.

<4> Section 2.2.2.2: When a client makes a request, it MUST specify the lowest capability version it can support as the *MinimumVersion* parameter. The client MUST specify the highest capability version it can support as the *MaximumVersion* parameter. The client MUST make the request in accordance with the *MaximumVersion* capability version.

When the server receives a request, it MUST compare its capability version to the capability version range the client presents. The server algorithm MUST assume that the client always makes a maximum-version request. The server MUST reject the request with an error if the *MaximumVersion* is higher than its highest capability version.

When the server responds to the client, including when it responds with an error, it MUST specify the lowest capability version it can support as the *MinimumVersion* parameter. The server MUST specify the highest capability version it can support as the *MaximumVersion* parameter.

If the server's maximum capability version is lower than the client's maximum capability version, the client SHOULD resend its request and alter its request to conform to the capability version range the server specified.

<5> Section 2.2.3.1.1.1.1: Windows uses a one-way hash of various machine characteristics to generate a HID. An example of machine characteristics includes the network address.

<6> Section 2.2.4.2.1.1: The QuotaResponse structure is kept in the protocol for backward compatibility but is not used. The CurrentConsumption member of this structure is set to 5 by the current server implementation. The Maximum member of this structure is set to 10 by the current server implementation. The Verified member of this structure is set to true.

<7> Section 2.2.5.2.1.1: The **ReferenceCertificates** response parameter is always returned as an empty value.

<8> Section 3: The RMS: Client-to-Server Protocol retains configuration information and RAC key data.

<9> Section 3.1.1: RMS server contains the public key of the SPC CA and checks that this key appears in the second or third certificate in the chain when validating SPC chains.

<10> Section 3.1.1: RMS server currently generates a random 1,024-bit RSA key pair on installation and retains this state.

<11> Section 3.1.3.1: All versions of RMS server earlier than version 2 contacted the Microsoft enrollment service to sign the SLC key into the hierarchy. The RMS version 2 server ships with a shared enrollment private key and certificate chain. When the RMS version 2 server initializes, it generates its own unsigned SLC, signs it with this shared enrollment private key, and appends the certificate chain.

<12> Section 3.1.5: RMS 1.0 SP2 client and later and RMS 2.0 server and later support Microsoft Web Browser Federated Sign-On authentication, as specified in [MS-MWBF].

<13> Section 3.1.6: RMS 1.0 SP2 client and later and RMS 2.0 server and later support Microsoft Web Browser Federated Sign-On authentication, as specified in [MS-MWBF].

<14> Section 3.1.7.2: The RMS server uses NTLM authentication through Microsoft Internet Information Services (IIS) for **Certify** requests.

<15> Section 3.1.7.2: All versions of RMS support NTLM authentication. RMS 2.0 server and later support Microsoft Web Browser Federated Sign-On authentication, as specified in [MS-MWBF].

<16> Section 3.1.7.2: The RMS server generates a unique 1,024-bit RSA key pair when a given user requests a RAC for the first time. This key pair is stored in a database for subsequent Certify requests from the same user. Subsequent requests reuse the same key pair to enable the roaming of ULs across multiple machines or devices.

<17> Section 3.1.7.3: The RMS server uses NTLM authentication through IIS for **FindServiceLocationsForUser** requests.

<18> Section 3.1.7.4: The RMS server generates a unique 1,024-bit RSA key pair each time it generates a CLC. This key pair is not stored on the server.

<19> Section 3.2.3.2.1: The RMS client checks the following string values in the Windows registry for server locations:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDRM\ServiceLocation]
"EnterprisePublishing"=
   [URL of server used for publishing and licensing]
"Activation"=[URL of server used for the Certify request]
```

<20> Section 3.2.4.3: These operating systems come with a Windows Management Instrumentation (WMI) job that can be enabled within an organization. The WMI job's template acquisition frequency is configurable through a group policy. When the WMI job is invoked, it invokes the RMS client functionality previously explained.

8 Index

U

V

W