[MS-RMSOD]:
Rights Management Services Protocols Overview

This document provides an overview of the Rights Management Services Protocols Overview Protocol Family. It is intended for use in conjunction with the Microsoft Protocol Technical Documents, publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A Protocol System Document does not require the use of Microsoft programming tools or programming environments in order to implement the Protocols in the System. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Abstract

This document provides an overview of the functionality and relationship of the Rights Management Services (RMS) protocols, which are specified in [MS-RMPR], [MS-RMPRS], and [MS-RMSI]. The Rights Management Services (RMS) protocols enable information-protection functionality that works with RMS-enabled applications to help safeguard digital information from unauthorized use, online and offline, inside and outside of the firewall. RMS is designed for organizations that need to protect sensitive and proprietary information, such as financial reports, product specifications, customer data, and confidential email messages.

This document describes the intended functionality of the Rights Management Services protocols and how these protocols interact with each other. It provides examples of some common use cases. It does not restate the processing rules and other details that are specific for each protocol. Those details are described in the protocol specifications for each of the protocols and data structures that belong to this protocols group.

Revision Summary

Date	Revision History	Revision Class	Comments
11/19/2010	1.0	New	Released new document.
01/07/2011	2.0	Major	Significantly changed the technical content.
02/11/2011	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	3.0	Major	Significantly changed the technical content.
09/23/2011	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	4.0	Major	Significantly changed the technical content.
03/30/2012	5.0	Major	Significantly changed the technical content.
07/12/2012	5.0	No change	No changes to the meaning, language, or formatting of

Date	Revision History	Revision Class	Comments
			the technical content.
10/25/2012	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	6.0	Major	Significantly changed the technical content.

Contents

1 Introduction

The Rights Management Services (RMS) protocols enable information-protection functionality that works with RMS-enabled applications to help safeguard digital information from unauthorized use, online and offline, inside and outside of the firewall. RMS is designed for organizations that need to protect sensitive and proprietary information, such as financial reports, product specifications, customer data, and confidential email messages. RMS can be used to help prevent sensitive information from intentionally or accidentally getting into the wrong hands.

This document describes the intended functionality of the RMS protocols and how the protocols in the RMS system interact. This includes how the RMS system interacts with systems or applications that create or consume rights-protected content, how RMS system servers interact, and how the RMS system interacts with management clients that need to configure and manage the system. It provides examples of some of the common user scenarios. It does not restate the processing rules and other details that are specific for each protocol. These details are described in the protocol specifications for each of the protocols and data structures that make up this system.

1.1 Conceptual Overview

This section summarizes concepts specific to this system, including:

Access and usage restrictions: The RMS system allows individuals and administrators to encrypt and specify access and usage restrictions on various types of data, including documents and email messages. This helps prevent sensitive information from being accessed and used by unauthorized people. This system includes persistent usage **policies**, and interacts with systems or applications that create or consume rights-protected content.

Persistent usage policies: The RMS system enhances an organization's security strategy by providing protection of information through persistent usage policies. This also enables usage rights, such as those restricting copying, printing, or forwarding, to be enforced after the information is accessed by an authorized recipient. RMS also helps organizations enforce corporate policy governing the control and dissemination of confidential or proprietary information. After permissions to content have been restricted by using rights management, the access and usage restrictions tend to be enforced no matter where the information is, because the access and usage restrictions tend to be stored in the content itself.

Activation and enrollment: Activation allows a machine to enroll in the RMS system. This process allows that machine to be used to create or consume usage policies. Servers can also be **decommissioned**, ending their participation in the RMS system.

Publishing and consuming protected content: The RMS protocols include functionality for publishing and consuming protected content. Content can be published online or offline.

ISV Extensions: These extensions provide a means for implementations to access the RMS system without necessarily going through the RMS client.

1.2 Glossary

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

certificate
certificate chain
forest
globally unique identifier (GUID)

NT LAN Manager (NTLM) Authentication Protocol
policy

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

client licensor certificate (CLC) chain
cloud service
consumer
content key
creator
license
offline publishing
online publishing
publishing license (PL)
rights policy template
RMS account certificate (RAC)
security processor certificate (SPC)
security processor certificate (SPC) private key
server licensor certificate (SLC)
service connection point (SCP)
use license (UL)

The following terms are specific to this document:

bootstrapping: The process RMS clients can use to self-activate.

1.3 References

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

[FIPS180-2] FIPS PUBS, "Secure Hash Standard", FIPS PUB 180-2, August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[MS-ADOD] Microsoft Corporation, "Active Directory Protocols Overview".

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

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

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

[MS-NTHT] Microsoft Corporation, "NTLM Over HTTP Protocol".

[MS-RMPR] Microsoft Corporation, "Rights Management Services (RMS): Client-to-Server Protocol".

[MS-RMPRS] Microsoft Corporation, "Rights Management Services (RMS): Server-to-Server Protocol".

[MS-RMSI] Microsoft Corporation, "Rights Management Services (RMS): ISV Extension Protocol".

[MSKB2627272] Microsoft Corporation, "AD RMS update to increase key lengths", http://support.microsoft.com/kb/2627272

[RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, http://www.ietf.org/rfc/rfc1945.txt

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

[RFC3377] Hodges, J., and Morgan, R., "Lightweight Directory Access Protocol (v3): Technical Specification", RFC 3377, September 2002, http://www.ietf.org/rfc/rfc3377.txt

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., et al., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

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

[SOAP1.2-2/2003] Gudgin, M., Hadley, M., Mendelsohn, N., et al., "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

[XML1.0] Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, October 2000, http://www.w3.org/TR/2000/REC-xml-20001006

[XMLNS-2ED] World Wide Web Consortium, "Namespaces in XML 1.0 (Second Edition)", August 2006, http://www.w3.org/TR/2006/REC-xml-names-20060816/

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

[XMLSCHEMA2] Biron, P.V., and Malhotra, A., Eds., "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 Version 1.2", 2001.

Note Contact the owner of the XrML specification for more information

2 Functional Architecture

2.1 Overview

Section 1, "Introduction", describes this document. This section introduces the system that is being documented.

The RMS system has three major roles: the **creator**, the **consumer**, and the **server**. The creator builds content and chooses an access policy for that content. When the RMS creator protects the content, it is encrypted by using a randomly generated **content key**. Both this key and the access policy are bound to the content in the form of a **publishing license (PL)**.

The consumer, upon receiving the document from the creator and opening it, supplies the server with the PL and the consumer's identity. If the consumer is allowed access, according to the access policy in the **license**, 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 identity.

A client (or an ISV extension application) can play the role of a creator, a consumer, or both, depending on the type of implementation. The client is responsible for requesting **certificates**, licenses, and policies from the server. The client is also responsible for enforcing authorization policies as they apply to protected information and encrypting or decrypting content as appropriate.

The server role in the RMS system 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 also responsible for evaluating and issuing authorization policies based on identity credentials that the client provides in protocol requests.

2.1.1 System Purpose

The RMS system provides the ability to secure information and restrict access to authorized users. It also provides the ability to enforce access policies and restrict information access to trusted applications. RMS provides an administration interface to manage the system and to create access policies. Figure 1 shows the major components that interact with the RMS system.

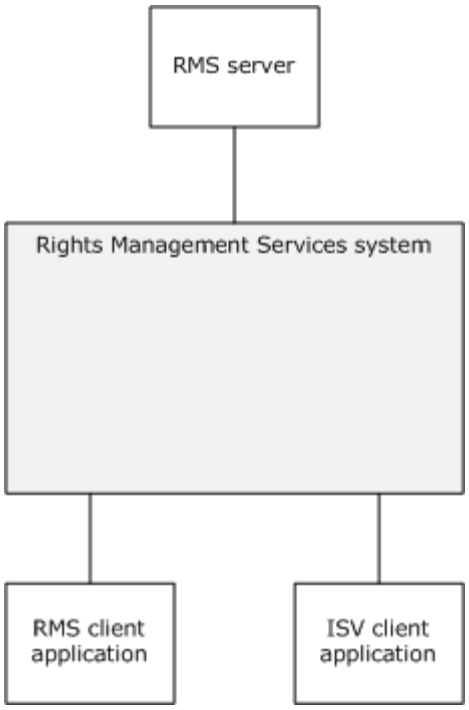

Figure 1: Major components interacting with the RMS system

The purpose of the RMS system is to:

- Provide a protection mechanism to ensure that data, such as email and documents, can be protected and consumed only by the intended recipients.
- Provide a record of transactions performed in the course of achieving this purpose.

The RMS system is comprised of the following functions:

- Issue the requisite certificates necessary for securing and consuming data.
- Store and manage the certificates previously mentioned.
- Store and manage **rights policy templates** (also known as "templates"), which provide policies for secured data defined on the enterprise level.

Creators use the system for the following purposes:

- Securing data, which includes the access policies for the data.
- Confirming content consumer identity and supplying the consumer with decryption keys and access rights.

Consumers use the system for the following purpose:

- Obtaining licenses to access protected data.

2.1.2 Functional Overview

This section describes the relationships between the system and external components, system dependencies, and other systems influenced by this system.

2.1.2.1 Abstract Components

At a high level, there are three main components in the RMS system: the RMS server, the RMS client application, and the ISV client applications. The RMS server, which acts only within the RMS system, provides RMS services such as issuing certificates and licenses. The RMS client application taps into the RMS system, acting as a client to an RMS server so that the user can protect content and access protected content. The ISV client application takes advantage of ISV extensions that supplement the client-server interface, providing direct access to the server for more advanced scenarios. ISV client applications can be implemented as standalone software, or as part of a system with RMS client implementations, as indicated by the implementation-specific connection in figure 2 (as follows). Figure 2 illustrates communication between the roles of the RMS system.

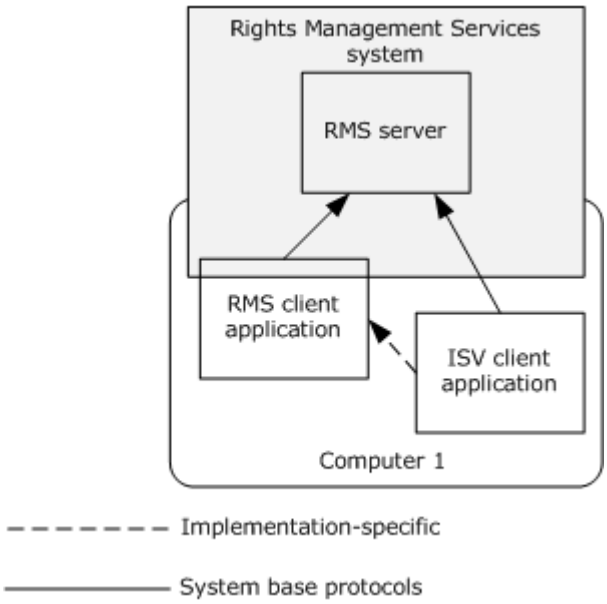

Figure 2: Basic communication with the RMS system

The following diagram illustrates how the RMS server exposes sets of interfaces that RMS client applications contact to perform RMS actions.

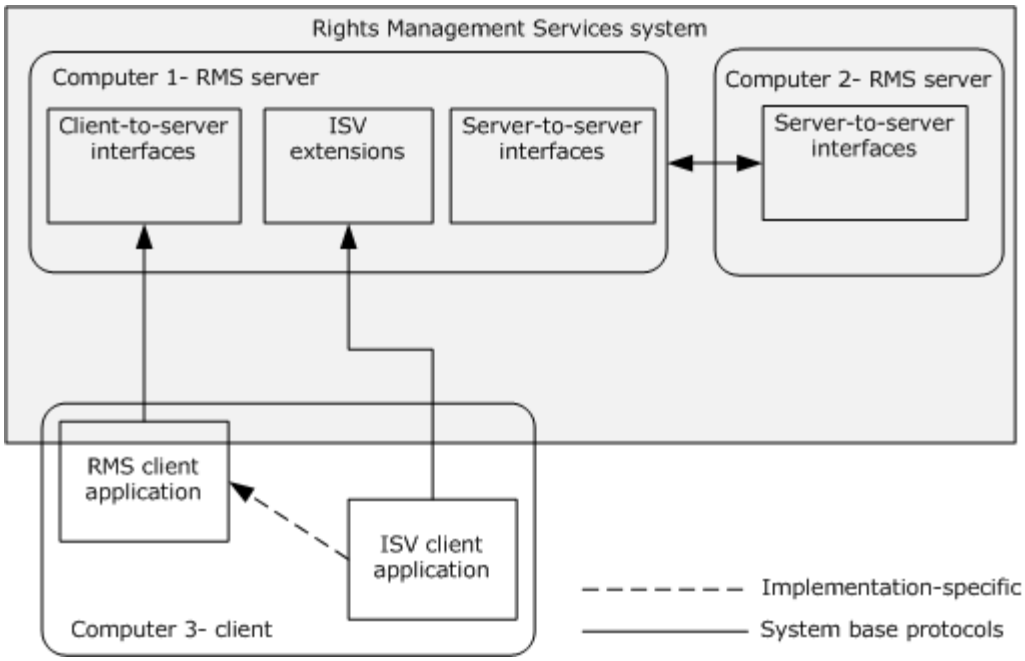

Figure 3: Distributed communication within the RMS system

The RMS server maintains RMS configuration, logging, and directory services databases. Additionally, certificates such as RMS account certificates are stored in the database. A directory service is also required to provide user authentication and other directory services.

2.1.2.2 Client-to-Server and Server-to-Server Functionality

The RMS protocols have two main sets of services, the client-to-server services and server-to-server services. The client-to-server services are the individual interfaces that RMS client applications call to perform RMS tasks such as **bootstrapping**, retrieving templates, publishing content, and licensing content. The server-to-server services are the system tasks performed to determine group membership of users across complex network directories. The RMS protocols also have an ISV extension interface that can be used to enable RMS in applications that do not necessarily use the RMS client services.

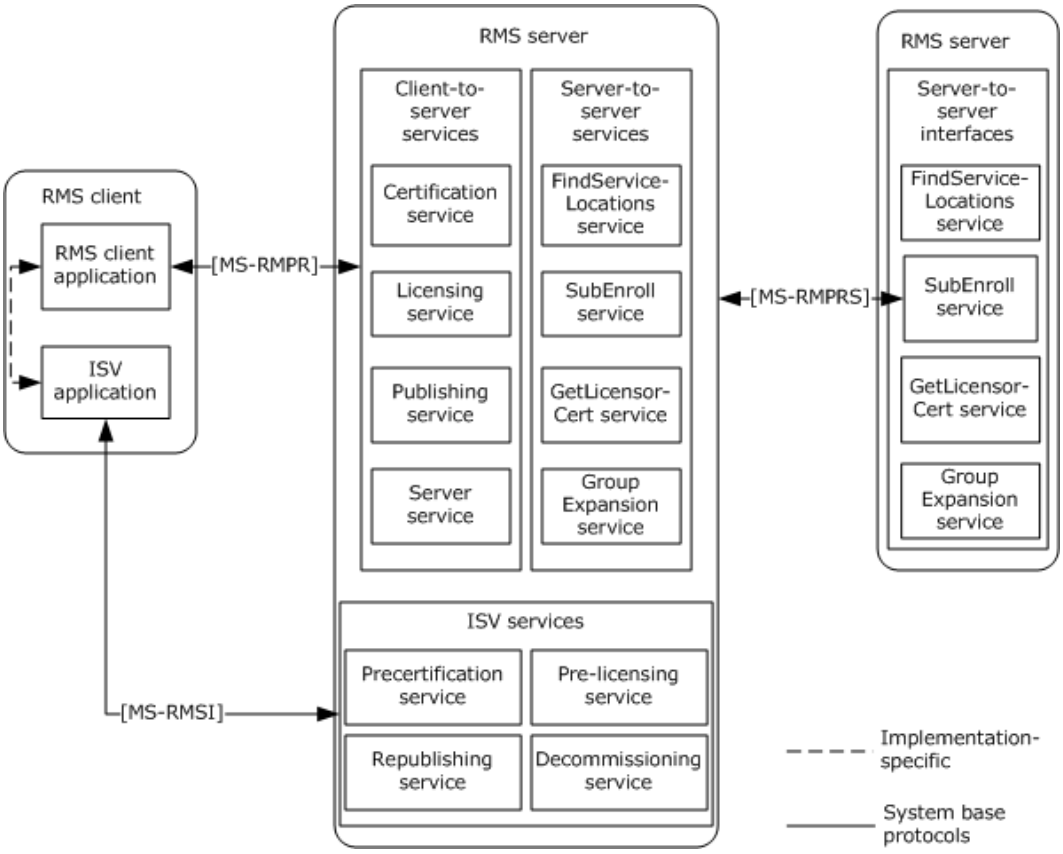

Figure 4: RMS white box diagram

Client-to-Server Interfaces

Certification Service: The certification service provides RMS client applications the ability to bootstrap and receive the necessary certificates to participate as a consumer in the RMS system. The certification service is the interface that RMS client applications use in the use case Bootstrap RMS Client - RMS Client Application detailed in section 2.5.4.2.

Certify: As part of the bootstrapping process, an RMS client makes a request to the Certify web method. This request includes the client's **security processor certificate (SPC)**; authentication is also performed as part of this request. The server validates the data it receives and if authorized returns the user's **RMS account certificate (RAC)**. Full details of the Certification Service schema can be found in [MS-RMPR] section 3.3. Details of the Certify method can be found in [MS-RMPR] section 3.3.4.1.

Licensing Service: The licensing web service provides two functions, issuing use licenses for protected content and providing rights policy templates to client applications that request them. Full details of the Licensing Service schema can be found in [MS-RMPR] section 3.4.

Acquire License: To consume protected content, the client needs to acquire a use license, which gives the user access to the content and includes the key to decrypt the content and the usage policies for the content. To acquire the use license the client uses the AcquireLicense method, sending their RAC, PL, and application data to the RMS server. After validating the RAC and PL in the request, the server returns the use license for the content. Details of the AcquireLicense method can be found in [MS-RMPR] section 3.4.4.1.

AcquireTemplateInformation: To retrieve rights policy templates, the client first makes a request to the AcquireTemplateInformation method. The server returns information about the available templates in the form of a list of **globally unique identifiers (GUIDs)** and hashes corresponding to the server templates. Details of the AcquireTemplateInformation method can be found in [MS-RMPR] section 3.4.4.2.

AcquireTemplates: After receiving the list of templates, the client determines which templates it needs to download from the server. The client uses the AcquireTemplates method with a list of rights policy template GUIDs and request templates corresponding to these GUIDs. Upon receiving the request, the server will return the templates to the client. Details of the AcquireTemplates method can be found in [MS-RMPR] section 3.4.4.3.

Publishing Service: The Publishing Service has two functions, to sign publishing licenses (PLs) generated during **online publishing** and to provide **client licensor certificate (CLC) chains**, which are used in **offline publishing**. The publishing service provides RMS client applications the ability to participate as a publisher in the RMS system. Full details of the Publishing Service schema can be found in [MS-RMPR] section 3.5.

AcquireIssuanceLicense: During online publishing, the RMS client application gets the **server licensor certificate (SLC)** chain from the server (see details on GetLicensorCertificate in section 3.3.2.1), generates a symmetric content key, and generates usage restrictions. The RMS client application generates a publishing license (PL), which includes the content key and use restrictions. The content key and use restrictions are encrypted with the server's public key (from the SLC).

In online publishing, after the PL is created, it will be signed by the server. The AcquireIssuanceLicense request is used to sign a PL during online publishing. The RMS client application makes an AcquireIssuanceLicense request to the server with the unsigned PL. The server signs the body of the PL and returns it to the RMS client application. Details of AcquireIssuanceLicense can be found in [MS-RMPR] section 3.5.4.1.

GetClientLicensorCert: During offline publishing, the RMS client application uses the CLC to publish content without contacting the RMS server. The CLC contains the server's public key, which is used to encrypt the content key and use restrictions in the PL. The CLC private key is used to sign the body of the PL.

The GetClientLicensorCert method is used to obtain the CLC for the user. The user needs a RAC and SPC to use this method. In the GetClientLicensorCert request, the client submits a RAC chain and requests a CLC chain. When the server receives the request, it performs signature validation on the RAC chain and verifies that it trusts the RAC. The server generates a CLC, which contains a unique asymmetric key pair. The server encrypts the CLC private key with the public key of the RAC, so the RAC and SPC are required to access the signing key in the CLC. Details of GetClientLicensorCert can be found in [MS-RMPR] section 3.5.4.2.

ServerService: The Server Service has two purposes, to provide the service location for users and to provide the SLC for use in online publishing. Full details of the Server Service schema can be found in [MS-RMPR] section 3.7.

FindServiceLocationsForUser: Depending on the RMS server configuration, different servers can be used for different functions for a given user. The client uses the FindServiceLocationsForUser request to discover the appropriate server for various services for a given user. The server uses the GetAuthenticatedAccount abstract interface to determine the authenticated domain account. When servicing FindServiceLocationsForUser requests, Windows implementations of the RMS server use the GetAuthenticatedAccount abstract interface to retrieve the domain account, which is authenticated using the **NT LAN Manager**

(NTLM) Authentication Protocol through Internet Information Services (IIS), as per [MS-NTHT]. The SOAP request does not encapsulate the authentication.

When the RMS client application makes a FindServiceLocationsForUser request, it includes a service type (for example, "user certification") and requests its location. Given the authenticated domain account and the requested server type, the server determines the service location using the GetDirectoryForAccount and GetServiceLocationForDirectory abstract interfaces and returns the URL to the RMS client application. Details of FindServiceLocationsForUser can be found in [MS-RMPR] section 3.7.4.2.

GetLicensorCertificate: During online publishing, the RMS client application retrieves the SLC from the RMS server. The SLC public key is used to encrypt the symmetric content key and use restrictions in the PL.

The RMS client application makes a GetLicensorCertificate request to the server; no information is sent in the request. Upon receiving the request, the server returns its SLC chain. Details of GetLicensorCertificate can be found in [MS-RMPR] section 3.7.4.1.

Server-to-Server Interfaces

Find Service Locations: RMS servers use the **Find Service Locations** interface of the RMS: Server-to-Server Protocol to find the URLs for specific services that are provided by other RMS servers. This communication can be useful in two scenarios:

1. Finding the appropriate URLs for a client to use for bootstrapping.
2. Finding the Group Expansion interface on a remote server before making a cross-**forest** group expansion request.

Clients contact the server for a bootstrapping process so they may begin functioning in the RMS system. This bootstrapping process is defined in the RMS: Client-to-Server Protocol, as specified in [MS-RMPR]. To bootstrap a specific user, the RMS server needs to authenticate that user and determine the user's email address by checking the directory. If the user's account resides in a separate directory (forest) that the RMS server cannot access, it cannot successfully bootstrap the user. A client starts the bootstrapping process by making a request for service locations to a specific RMS server. If that RMS server is not the appropriate server to bootstrap the client, the server can use the **Find Service Locations** interface to find the URLs on the appropriate server and then return them to the client.

The **Find Service Locations** interface uses a SOAP-based protocol over HTTP. It exposes one request/response method: **FindServiceLocations**.

Sub-Enrollment: RMS servers use the **Sub-Enrollment** interface of the RMS: Server-to-Server Protocol to bootstrap subordinate RMS servers.

As shown in Figure 4, an RMS server can be deployed as a subordinate to another RMS server. A root RMS server grants a subordinate RMS server the right to perform only licensing tasks by issuing a subordinate server licensor certificate (SLC) from its own. For a subordinate RMS server, this process replaces the standard RMS server bootstrapping process defined in [MS-RMPR] section 3.1.3.

The **Sub-Enrollment** interface uses a SOAP-based protocol over HTTP. It exposes one request/response method: **SubEnroll**.

Get Licensor Certificate: RMS servers use the **Get Licensor Certificate** interface of the RMS: Server-to-Server Protocol to establish trust from a root server to a subordinate server.

When a subordinate RMS server is deployed, it needs to trust identities issued by the root RMS server. This is accomplished by trusting certificates that were issued by the root RMS server by trusting the root RMS server's public key. The subordinate server can use the **Get Licensor Certificate** interface to retrieve the SLC of the main server that contains the appropriate public key.

The **Get Licensor Certificate** interface uses a SOAP-based protocol over HTTP. It exposes one request/response method: **GetLicensorCertificate**.

Group Expansion over SOAP: RMS servers use the **Group Expansion over SOAP** interface of the RMS: Server-to-Server Protocol to determine group membership of authorized users across complex network directories.

Access policy on RMS-protected content can specify individual users as well as distribution groups. When a consumer contacts the RMS server for authorization to access protected content, the server might need to consult the directory to determine whether that user is a member of a groups with rights granted to use the content. If that group exists in a separate directory (forest) to which the RMS server does not have access, the RMS server needs to contact another server that does have access to that directory and that can provide information about the group membership. This server-to-server communication can use either the **Group Expansion over SOAP** interface or the **Binary Group Expansion** interface.

The **Group Expansion over SOAP** interface exposes one request/response method: **IsPrincipalMemberOf**.

Binary Group Expansion: The Binary Group Expansion interface performs the same function as the **Group Expansion over SOAP** interface, however, it does so using a binary-formatted message, sent over HTTP. It exposes one request/response method: **IsPrincipalMemberOf**. This interface is not available in RMS version 2.0; for more details, see [MS-RMPRS] section 3.5.

ISV Extension Interfaces

Decommission server: If an organization were to decide to stop using RMS entirely and remove its deployment, it would need to remove RMS protection from content. One method is to have people with owner rights to each piece of content remove the protection. Realistically, however, it is often impractical to find these people since they may no longer belong to the organization in question. Another approach is to use the Decommissioning interface to extract the content key from a PL and return it so that it can then be used to decrypt the content. Because each protected document has a PL, and each PL has its own content key, this process will have to be repeated for each protected document that needs to have its protection removed.

When servicing the request, the RMS server does not verify that the requestor is supposed to be granted access to the content as indicated in the PL. Rather, the RMS server will return the content key to any requestor. As a result, the Decommissioning interface is disabled for normal operation by default. The interface exposes one request and response method to support decommissioning: **AcquireContentKey**.

Precertification: When protected content is sent to recipients, each recipient acquires a use license that grants access to the content. The use license describes the usage policy for that user with that content and encrypts the content key to the user's public key. This process and protocol is described in the RMS: Client-to-Server Protocol Specification [MS-RMPR].

As an optimization, the use license for a recipient could be generated in advance and made available with the content at the time the recipient attempted to access it. The use license

could be requested on behalf of the recipient by either the sender or a server application that might be involved in delivering the content to the recipient. This use license would allow the recipient to access the content as soon as it was delivered without having to contact the RMS server, presuming that the recipient has already been bootstrapped.

In order to acquire a license on behalf of a recipient user, a requestor retrieves the public part of the recipient's RMS account certificate (RAC) using the Precertification interface and then request a use license from the RMS server using the RMS: Client-to-Server Protocol [MS-RMPR]. The Precertification interface exposes one request and response method to enable precertification: **Precertify**.

Republishing: After protected content is published, it might become necessary to alter the set of rights that are granted to users in the original PL. The EditIssuanceLicense operation ([MS-RMSI] section 3.4.4.1) allows a client to submit the original signed PL, as well as an unsigned PL that contains the altered rights. The RMS server responds with a new signed PL that contains the same content key as the original PL.

PLs are required to permit republishing, as described in [MS-RMSI] section 3.4.4.1. In addition, access to this service is typically restricted to computers or users trusted by the administrator. The Republishing interface exposes one request and response message to enable republishing via the EditIssuanceLicense operation.

Prelicensing: When using the Precertification interface, the application is required to contact a server that is capable of issuing an RAC for a specific recipient. In an environment with multiple certification services, an application might require an application-specific configuration to determine which certification service to use for each user. If multiple applications prelicense content, an administrator might have to configure this data in independent ways.

The Prelicensing interface shifts this responsibility to the RMS server. The application can specify a list of recipients by email address and provide a PL. The RMS server will determine the public key for each user and issue a use license for each recipient based on the rights granted in the PL. The RMS Server itself can use the Precertification interface of another server to retrieve a public key for a user when their key resides on that server. The Prelicensing interface exposes one request and response message to enable prelicensing via the AcquirePreLicense operation ([MS-RMSI] section 3.5.4.1).

2.1.3 Communication Within the System

The RMS protocols use the SOAP messaging protocol for all communication. Refer to [MS-RMPR], [MS-RMPRS], and [MS-RMSI] for full details on implementation of these interfaces.

2.1.4 Applicability

Applicability: The RMS protocols are used to manage user access to protected information, such as documents, email, and files.

2.1.5 Relevant Standards

Conformance with external standards: The RMS protocols use the following standards to allow interoperability with other external systems.

Hypertext Transfer Protocol, as specified in [RFC2616]. This protocol provides a standard for clients and servers to communicate. It defines how messages are formatted and transmitted, and what actions web servers and client applications take in response to various commands.

Extensible Markup Language, as specified in [XML1.0], [XMLNS-2ED], [XMLSCHEMA1], [XMLSCHEMA2], [XPATH]. This standard provides a format for describing structured data. This facilitates more precise declarations of content and more meaningful search results across multiple platforms.

Secure Hash Standard, as specified in [FIPS180-2]. SHA hashes are used to sign messages. The RMS protocols can use SHA-1 or SHA256 hashes to sign messages, as specified in [MS-RMPR] sections 2.2.9.1.12 and 3.1.4.7.<1>

eXtensible Rights Markup Language, as specified in [XRML]. This standard provides descriptions of usage rights and conditions for digital contents, together with message integrity and entity authentication in these descriptions.

SOAP, as specified in [SOAP1.1], [SOAP1.2-1/2003], [SOAP1.2-2/2003]. This standard provides a standard Internet protocol for exchanging structured information in a distributed environment.

Web Services Description Language, as specified in [WSDL]. This standard provides a general purpose XML language for describing the interface, protocol bindings, and the deployment details of network services.

NTLM over HTTP, as specified in [MS-NTHT]. This protocol provides network logon authentication using a challenge-response process.

The Active Directory system described in [MS-ADOD] can be used with the Group Expansion services.

The LDAPv3 protocol, as specified in [RFC3377]. This can be used to retrieve the service connection points from Active Directory.

Microsoft Web Browser Federated Sign-On Protocol, as specified in [MS-MWBF] and [MS-MWBE]. This standard allows integration with Active Directory Federation Services to support Federated Identities. It enables two-way collaboration between organizations when only one organization has an RMS server.

2.2 Protocol Summary

The table below provides a comprehensive list of the RMS member Protocols.

Protocol name	Description	Short name
Rights Management Services (RMS): Client-to-Server Protocol	This protocol obtains and issues certificates and licenses used for creating and working with protected content. [MS-RMPR] is used for all client and server communication in relation to bootstrapping a client, consuming protected content, or protecting content. This protocol uses SOAP ([SOAP1.1] or [SOAP1.2-1/2007]) over HTTP [RFC2616]. HTTPS support is recommended, however, it is only required when using the Microsoft Web Browser Federated Sign-on Protocol [MS-MWBF] in conjunction with the Licensing or Certification interfaces.	[MS-RMPR]
Rights Management Services (RMS): Server-to-Server Protocol	This protocol locates RMS services, enrolls servers in the RMS system, and performs group expansion. [MS-RMPRS] is used for communication between RMS servers in relation to finding service locations, group expansion, and enrolling	[MS-RMPRS]

Protocol name	Description	Short name
	subordinate RMS Servers. This protocol uses SOAP over HTTP. HTTPS support is recommended, but not required. Binary group expansion uses the HTTP transport.	
Rights Management Services (RMS): ISV Extension Protocol Specification	This protocol communicates information between applications and RMS servers directly without using the RMS client.<2> [MS-RMSI] is used for scenarios that are not supported by the RMS: Client-to-Server Protocol in relation to decommissioning protected content, precertifying users, prelicensing content, and republishing content. This protocol uses SOAP over HTTP. HTTPS support is recommended, but not required.	[MS-RMSI]

2.3 Environment

The following sections identify the context in which the system exists. This includes the systems that use the interfaces provided by this system of protocols, other systems that depend on this system, and, as appropriate, how components of the system communicate.

The RMS protocols require network connectivity between clients and servers using the HTTP protocol, typically over port 80. Optionally HTTPS can be used, typically over port 443. The network needs to have Domain Name Services (DNS) with the RMS server registered in DNS. Clients can be located on private networks, such as an enterprise network managed by an IT department, or can communicate over the Internet.

An RMS client can be any device with the capability to connect to an RMS server using HTTP. Client implementations are free to persistently store certificates provided by the RMS server.

An RMS server is a web server capable of communicating with clients over HTTP.

2.3.1 Dependencies on This System

There are no systems that depend on the RMS protocols.

2.3.2 Dependencies on Other Systems/Components

The RMS protocols depends on the following technologies:

- SOAP (section 2.3.2.1)
- Cryptographic keys (section 2.3.2.2)
- Directory services (section 2.3.2.3)
- Federated Sign-On (section 2.3.2.4)
- XrML (section 2.3.2.5)
- RMS certificates and licenses (section 2.3.2.6)

2.3.2.1 SOAP

SOAP is a simple XML-based protocol that enables applications to exchange information over HTTP. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

The RMS protocols use 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 to be the wire format used for messaging, and HTTP and HTTPS are the underlying transport protocols. This requires that clients and servers have network connectivity and are properly configured to use TCP/IP. There is no specific requirement for the type of physical networking topology. For more information on SOAP, see [SOAP1.1], [SOAP1.2-1/2003], and [SOAP1.2-2/2003].

2.3.2.2 Cryptographic Keys

The RMS system uses both symmetric and asymmetric (also known as public-key) cryptography. Cryptography in RMS is used to protect various certificates, licenses, and content. This provides organizations with a seamless way to protect and unprotect content. without requiring their users to have any knowledge of the underlying system.

Symmetric-key cryptography refers to encryption methods in which the key that was used to encrypt information is the same key that decrypts the information. In asymmetric cryptography there are two keys, a public key and a private key. The keys are mathematically related but it is not computationally feasible to determine one key with only the other. The public key can be freely distributed and is generally used to encrypt data or verify signatures. The private key is kept secret and is generally used for decrypting and signing data.

2.3.2.3 Directory Services

RMS uses directory services, such as Active Directory, as a central repository for storing and retrieving identity and account information about RMS users. The directory services are also used to enable the RMS client to discover the RMS server and authenticate requests to the server. In scenarios where Active Directory is used as the directory service and the RMS server has joined a domain, the domain serves as the primary source of identity for the RMS server and RMS users. The domain, through the relevant security protocols, provides the basis for authentication within the domain, allowing principals within the domain to establish authenticated connections with each other. Once authenticated, the domain provides authorization information in the form of additional identities representing groups, whereby authorization decisions may be made. RMS identifies users by email addresses (or SIDs), which are stored in the directory. In scenarios where other directory services are used, separate authentication mechanisms can be used, such as anonymous authentication. See [MS-ADOD] for more information on the Active Directory system.

2.3.2.4 Federated Sign-On

Use of Federated Sign-On allows enterprises to establish relationships for exchanging protected content with entities outside their directory infrastructure. For more information on Federated Sign-On, see [MS-MWBE] and [MS-MWBF].

2.3.2.5 XrML

The eXtensible rights Markup Language (XrML) is a general-purpose, XML-based specification grammar for expressing rights and conditions associated with digital content, services, or any digital resource.

The RMS system uses an XML vocabulary to express digital rights: the eXtensible rights Markup Language (XrML), version 1.2.1. XrML specifies a rights-expression language that trusted systems in a trusted environment can use to express digital information policies. You can apply XrML licenses to trusted information that is in any format, such as email, office productivity tools, database contents, e-commerce downloads, line-of-business programs, and customer relationship management systems, to name a few. You can then enforce XrML licenses through any trusted rights management system that uses the XrML standard.

The rights to be managed are expressed in an XrML PL that is associated with the protected information. The PL is an expression of how the information owner wants it to be used, protected, or distributed. The PL and the user's identity are passed to the RMS system, and if there is a direct correlation between the user's identity and the rights specified in the PL, the RMS system returns a use license.

These licenses are easily interpreted and managed by various interoperable rights management systems because they all use the XrML standard. Using licenses to manage information online provides ease-of-access from any location. After the use license is downloaded, a client implementation will be able to access protected information both online and offline.

XrML supports an extensive list of rights. In addition, applications can define additional rights to meet particular needs. By defining additional rights, enterprises can build many business, usage, and workflow models to meet their specific requirements. For more information on XrML, see [XRML].

2.3.2.6 RMS Certificates and Licenses

RMS defines specific XrML certificates to identify and trust different entities in the system. Licenses are also XrML certificates but are used to specify rights and conditions that govern content use. The following sections detail the certificates and licenses that are used by the RMS system.

Server licensor certificate: The SLC represents a root of trust in the system and the enterprise. It is the identity of an RMS server, and enables a server to issue certificates and licenses for working with protected content. The SLC grants the right to issue:

- Publishing licenses
- Use licenses
- Client licensor certificates
- Rights policy templates
- Rights account certificates

security processor certificate: The security processor certificate (SPC) is generated during activation and contains the public key corresponding to the **security processor certificate (SPC) private key**. The SPC represents the identity of a computer that can be used for working with protected content. For more information on activation, see [MS-RMPR] section 3.8.4.1.

RMS account certificate: The RMS account certificate (RAC) represents the identity of a user who can access protected content.

client licensor certificate: The client licensor certificate (CLC) enables a user to publish protected content offline.

Publishing license: The publishing license (PL) 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 PL tells the server what usage policies apply to a given piece of content and grants the server the right to issue use licenses based on that policy. The PL is created when content is protected.

Use license: The use license (UL) authorizes access to a given piece of protected content and describes the usage policies that apply. The UL contains the symmetric content key for decrypting the content.

2.4 Assumptions and Preconditions

Given the environment described in section 2.3, the system has the following assumptions and preconditions:

- The RMS server requires a database and stored procedures to perform operations.
- The RMS server has a directory available to authenticate users and retrieve the email address of user accounts. All user accounts and groups who use RMS to consume and publish content have an email address that is configured in the directory.
- DNS registration is configured for the RMS servers.
- Each RMS Web service supports SOAP [SOAP1.1] over HTTP [RFC2616] over TCP/IP.

Member protocols supported by the system, as listed in section 2.2, can have additional assumptions and preconditions when that protocol is being used. Please see the relevant member protocol specification for details.

2.5 Use Cases

This section specifies the major use cases of the RMS protocols and the rationale for their use.

2.5.1 Actors

Stakeholders (actors) that use the RMS protocols include users, computers, applications, servers, and services. The actors that participate in the RMS use cases are:

RMS user: A person who uses an RMS client Application. The primary interests of an RMS user are to be able to protect and consume protected content.

Client computer: A computer or device, such as a mobile phone, that hosts an RMS client application.

RMS client application: An application that acts as a client to an RMS server. The application can be an end user-based or server-based application, and can perform RMS functions such as protecting content and providing access to protected content.

ISV application: An application that may or may not utilize the RMS client. An application can be an end-user client application or a server application that utilizes RMS.

RMS server: The component that provides RMS services, such as issuing certificates and licenses.

RMS administrator: A person who performs RMS administration in the enterprise and typically has full access to RMS servers. The interests of the RMS administrator are to configure the RMS server for use in the system.

RMS cloud service: A Web service run by Microsoft, available on the Internet, which provides enrollment services to RMS servers.<3>

2.5.2 Supporting Actors and System Interests Summary

There are no other systems in which the RMS system is an actor.

2.5.3 Use Case Summary Diagrams

The following table provides an overview for the groups of use cases that span the functionality of the RMS protocols. The sections that follow provide detailed descriptions of the use cases in each group. Each use case is described in detail in section 2.5.4.

Use case group	Use cases
Prepare RMS clients and servers to participate in the RMS system	Enroll RMS Server - RMS Server Bootstrap RMS Client - RMS Client Application Sub-enroll Server - RMS Server
Protect content using RMS, consume content that has been protected using RMS	Find Service Locations for Client- RMS Server Acquire Templates - RMS Client Application Publish Protected Content Online - RMS Client Application Publish Protected Content Offline - RMS Client Application Consume Protected Content - RMS Client Application Expand Groups - RMS Server Find Service Locations for Group Expansion- RMS Server
Access the server for advanced scenarios	Decommission Server - ISV Application Republishing Content - ISV Application Perform Precertification - ISV Application Perform Prelicensing - ISV Application

The following diagrams illustrate the use cases in each use case group.

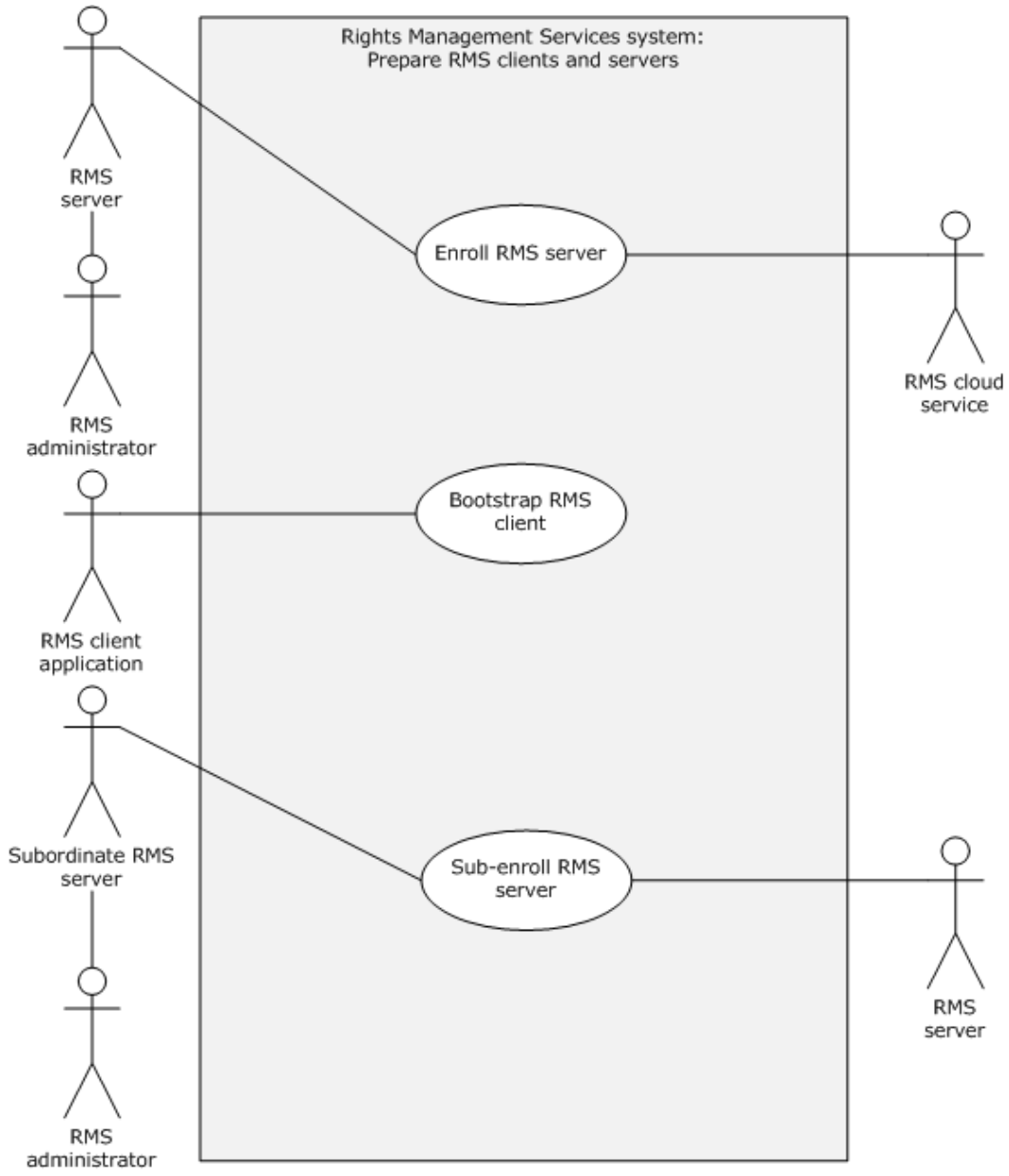

Figure 5: Preparing RMS clients and servers to participate in the RMS system

Figure 6: Protect content by using RMS; consume content that has been protected by using RMS

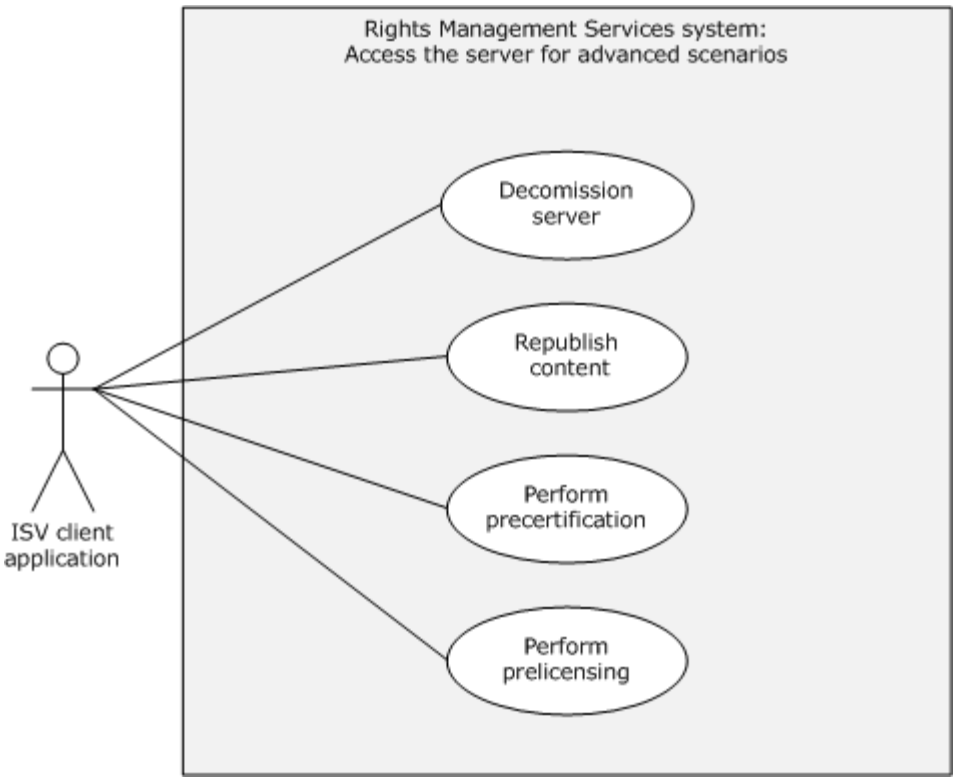

Figure 7: Access the server for advanced scenarios

2.5.4 Use Case Descriptions

2.5.4.1 Enroll RMS Server - RMS Server

Goal: Enroll the server with the Microsoft **cloud service** so clients trust the server and send it requests.

Context of use: An RMS server performs enrollment before servicing any client requests. Servers perform enrollment by generating an enrollment request and sending it to the RMS cloud service. Server enrollment requests contain the public portion of the RMS server's key pair and other enrollment information such as its GUID. Server enrollment requests can be made synchronously by the server directly contacting the RMS cloud service, or asynchronously by an RMS administrator exporting the enrollment request and contacting the RMS cloud service from another computer. For more information, see [MS-RMPR] section 3.1.3.2.

Direct actor: The direct actor of this use case is the RMS server.

Primary actor: The primary actor is the RMS administrator.

Supporting actors: The supporting actor is The RMS cloud service.

Stakeholders and interests:

- RMS server, as described in section 2.5.1.
- RMS administrator, as described in section 2.5.1.

- RMS cloud service, as described in section 2.5.1.

Preconditions: The server generates an asymmetric key pair for the certificate that will represent the server's identity.

Minimal guarantees: If a properly formatted enrollment request is sent to the RMS cloud service, a server certificate will be generated, signed, and appended to the server enrollment **certificate chain**.

Success guarantee: The minimal guarantee is the same as the success guarantee.

Trigger: The administrator triggers this use case after RMS is installed on a server and when the server's certificate needs to be renewed.

Main success scenario:

1. The RMS server makes an enrollment request to the RMS cloud service.<4>
2. The RMS cloud service returns a signed server certificate and the server enrollment certificate chain, which is then used by the server.

Extensions:

Enrollment requests can also be made asynchronously by an RMS administrator exporting the enrollment request and sending it to the RMS cloud service from another computer.

2.5.4.2 Bootstrap RMS Client - RMS Client Application

Goal: Prepare an RMS client application to participate in the RMS system.

Context of use: Client bootstrapping is a set of initialization steps that clients complete before performing either offline publishing or consuming content. During client bootstrapping, the client computer and RMS user are configured to participate in the RMS System. This involves various encryption key and certificate generations and exchanges.

Direct actor: The direct actor of this use case is the RMS client application.

Primary actor: The primary actor is the same as the direct actor.

Supporting actors: None.

Stakeholders and interests:

- RMS client application, as described in section 2.5.1.
- RMS user, as described in section 2.5.1.
- Client computer, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.

Preconditions: Ability to discover the RMS services.

Minimal guarantees: The RMS user and client computer can be uniquely identified as participants in the RMS system, and will receive the appropriate certificates (RAC) to consume protected content.

Success guarantee: The minimal guarantee, plus receiving the CLC, which grants the ability to publish offline content.

Trigger: Typically this use case is triggered by an RMS client application needing to protect or consume protected content for the first time. This can be initiated by an RMS user using an RMS client application, or by automation in an RMS client application.

Main success scenario:

1. The RMS client application discovers the RMS services necessary for this operation. This operation makes use of the certification service, and optionally, the publishing service.
2. The RMS client application generates a security processor certificate (SPC) that is client-computer specific.
3. The RMS client application sends the SPC to the certification RMS server and requests the user's RMS account certificate (RAC).
4. The certification RMS server validates the SPC and identity of the user, and sends the RMS client application the RAC.
5. To publish offline, the user needs to have a separate signing certificate that is bound to the user's identity in RMS. The client first finds the service location, by deriving it from a PL or discovering it from the directory service.<5> The client then sends a request to the publishing RMS server to retrieve the CLC.

Extensions: None.

2.5.4.3 Sub-Enroll Server - RMS Server

Goal: RMS servers use **Sub-Enrollment**, part of the RMS: Server-to-Server Protocol, to bootstrap subordinate RMS servers. RMS servers also use the **Get Licensor Certificate** interface of the RMS: Server-to-Server Protocol to establish trust from a root server to a subordinate server.

Context of Use: an RMS server can be deployed as a subordinate to another RMS server. A root RMS server grants a subordinate RMS server the right to perform only certain licensing tasks by issuing a subordinate SLC from its own. For a subordinate RMS server, this process replaces the standard RMS server bootstrapping process defined in [MS-RMPR] section 3.1.3.

When a subordinate RMS server is deployed, it needs to trust identities issued by the root RMS server. This is accomplished with a trusted certificate chain issued by the root RMS server with the root RMS server's trusted public key. The subordinate server can use the **Get Licensor Certificate** interface to retrieve the SLC of the main server that contains the appropriate public key.

Direct actor: Subordinate RMS server.

Primary actor: The primary actor is the RMS administrator.

Supporting actors: The root RMS server.

Stakeholders and interests:

- RMS administrator.
- Subordinate RMS server.
- Root RMS server.

Preconditions:

- The root RMS server needs to be enrolled in the RMS system and capable of granting SLCs.

- The subordinate RMS server needs to be able to contact the root RMS server.

Minimal guarantee: The subordinate RMS server can grant limited content licenses.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: The RMS administrator needs to create a subordinate RMS server on behalf of the organization.

Main success scenario:

1. The subordinate RMS server requests an SLC by sending a SubEnroll request to the root RMS server. This request includes a public key and the subordinate RMS server's attributes.<6>
2. The root RMS server responds with an SLC, granting the subordinate RMS server the right to be an RMS server.
3. The subordinate RMS server requests a limited licensor certificate from the root RMS server by using the GetLicensorCert operation.
4. The root RMS server returns a licensor certificate to the subordinate RMS server.

Extensions: None.

2.5.4.4 Find Service Locations for Client - RMS Server

Goal: Finding the appropriate URLs for an RMS client to use.

Context of use: RMS servers use the **Service Location** interface of the RMS: Server-to-Server Protocol to find the URLs for specific services that are provided by other RMS servers.

Direct actor: The direct actor of this use case is the specific RMS server the client has contacted.

Primary actor: The primary actor is the same as the direct actor.

Supporting actors: Any number of other RMS servers.

Stakeholders and interests:

- RMS client application, as described in section 2.5.1.
- RMS user, as described in section 2.5.1.
- Client computer, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.

Preconditions: The specific RMS server already has a list of other RMS servers in the system.

Minimal guarantees: The RMS server returns a URL for the requested service to the client.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: A client starts the process by making a request to the specific RMS server located at a **service connection point (SCP)** known to the client.

Main success scenario:

1. The specific RMS server receives a request from the client. This request includes one or more service types.

2. The specific RMS server sends a request to one or more servers on its list of other servers in the RMS system. The request identifies the type of service(s) requested.

3. The other RMS servers will each respond.

4. At least one of the responses includes a valid set of SCPs for the service(s) requested. This can be an SCP pointing to the other RMS server that responded, or an SCP that is known to that server.

5. The specific RMS server returns the SCP to the client.

Extensions: None.

2.5.4.5 Acquire Templates - RMS Client Application

Goal: Retrieve the rights policy templates published by the RMS server for use in publishing protected content.

Context of use: Rights policy templates contain a pre-determined access policy that can be used by an RMS client application to assign rights when publishing protected content. Rights policy templates are published on the RMS server and, in order to be used have to be retrieved by the RMS client application.<7>

Direct actor: The direct actor of this use case is the RMS client application.

Supporting actors: None.

Stakeholders and interests:

- RMS user, as described in section 2.5.1.
- RMS client application, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.
- Client computer, as described in section 2.5.1.

Preconditions: The RMS client application needs to be able to determine the location of the RMS server by the means specified in [MS-RMPR] section 3.8.3.2.

Minimal guarantees: The RMS server returns the complete set of rights policy templates. It can also return individual rights policy templates when requested. The client can store individual templates in a local license store.

Success guarantee: The set of templates that have been updated on the server have been propagated to the client.

Trigger: This scenario can be triggered at any time, but generally an RMS client application triggers this use case at regular intervals to ensure it has the latest rights policy templates. An RMS client application can also allow an RMS user to trigger this use case.

Main success scenario:

1. An RMS client application requests the list of templates available from the RMS server by using the AcquireTemplateInformation operation ([MS-RMPR] section 3.4.4.2).

2. The RMS client application makes subsequent requests to the server for individual rights policy templates by using the AcquireTemplates operation([MS-RMPR] section 3.4.4.3). The client places the templates from the server in a local license store.

Extensions: None.

2.5.4.6 Publish Protected Content Online - RMS Client Application

Goal: Publish protected content by communicating directly with the RMS server.

Context of use: Online publishing allows publishing content by acquiring the public portion of the RMS server's SLC, generating a PL for the content, and sending that license to the server to be signed. Online publishing does not require client bootstrapping.

Direct actor: The direct actor of this use case is the RMS client application.

Primary actor: In an end-user client application, the primary actor is the RMS user. The primary actor can also be the RMS client application, for example in the case of server or automated applications.

Supporting actors: None.

Stakeholders and interests:

- RMS user, as described in section 2.5.1.
- RMS client application, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.

Preconditions:

- If rights policy templates are used to protect the content, they need to have already been retrieved per section 2.5.4.4.
- The RMS client application needs to be able to determine the location of the RMS server by the means specified in [MS-RMPR] section 3.8.3.2.

Minimal guarantees: The RMS client application receives a signed PL for the content. The PL contains the location of the RMS server to retrieve a use license to consume the content. The PL allows the intended recipient to consume the content.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: An attempt to protect content using RMS, which can be trigged by an RMS user using an RMS client application, or by automation in an RMS client application.

Main success scenario:

1. The RMS client application sends a request to the RMS server to retrieve the public portion of the server's SLC.
2. The application generates a PL for the content being protected, which contains the content key and usage policy.
3. The application encrypts the content key and usage policy in the PL by using the SLC public key.

4. The client sends the PL to the RMS server to be signed, by using the AcquireIssuanceLicense method ([MS-RMPR] section 3.5.4.1).

5. The RMS Server returns the signed PL for the content.

Extensions: None.

2.5.4.7 Publish Protected Content Offline - RMS Client Application

Goal: Protect content without making calls to an RMS server.

Context of Use: Offline publishing gives an RMS client application the ability to publish content without making calls to the RMS server. Unlike online publishing, client bootstrapping is required for offline publishing.

Direct actor: The direct actor of this use case is the RMS client application.

Primary actor: In an end-user client application, the primary actor is the RMS user. The primary actor can also be the RMS client application, for example in the case of server or automated applications.

Supporting actors: None.

Stakeholders and interests:

- RMS client application, as described in section 2.5.1.
- RMS user, as described in section 2.5.1.

Preconditions:

- The client needs to be bootstrapped with the RMS server, per section 2.5.4.2, including having a CLC.
- If a template is used to publish offline, the client verifies that the key used to sign the template is the same as the key used to sign the CLC.

Minimal guarantees: The RMS client application protects the content and generates a publishing license for it. The publishing license contains the URL of the RMS server that can issue a use license for the content. The license for the protected content allows the intended recipient to consume the content.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: An attempt to protect content by using RMS, which can be trigged by an RMS user using an RMS client application, or by automation in an RMS Client application.

Main success scenario:

1. The RMS client application protects the content and generates a publishing license for the protected content, which contains the usage policy and the content key.

2. The usage policy and content key in the license are encrypted using the server's SLC public key, which was retrieved from the CLC.

3. The license for the content is signed using the CLC private key.

Extensions: None.

2.5.4.8 Consume Protected Content - RMS Client Application

Goal: Remove protection from and consume content protected by RMS.

Context of use: Upon receiving protected content, the RMS client application submits the publishing license and RAC to the server. The server validates the data received and returns a use license that is used to unprotect the content for consumption.

Note Opaque, binary application data may also be returned attached to the AcquireLicenseParams response. See [MS-RMPR] section 3.4.4.1.3.3 for more information.

Direct actor: The direct actor of this use case is the RMS client application.

Primary actor: In an end-user client application, the primary actor is the RMS user. The primary actor can also be the RMS client application, for example in the case of server or automated applications.

Supporting actors: None.

Stakeholders and interests:

- RMS user, as described in section 2.5.1.
- RMS client application, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.

Preconditions:

- The client needs to be bootstrapped with the RMS server, per section 2.5.4.2.
- The RMS client application needs to be able to determine the location of the RMS server by the means specified in [MS-RMPR] section 3.8.3.2.

Minimal guarantees: The RMS server will evaluate the publishing license, RAC, and application data. If it trusts all three, it will return the appropriate license; if it does not trust all three, it will return the appropriate fault code.

Success guarantee: The RMS client application receives the license to consume the content and is able to remove the protection.

Trigger: An attempt is made to access RMS-protected content using an RMS client application. This can be trigged by an RMS user using an RMS-aware application, or by automation in an RMS client application.

Main success scenario:

1. The RMS client application sends the publishing license that came with the content, the RAC, and the application data to the RMS server.
2. The RMS server validates the data received. If it trusts the PL and RAC, the server generates and sends a Use License for the RMS client application to consume the content.
3. The RMS client application receives the license to consume the content. The license contains the protected symmetric content key and usage policies.

Extensions: None.

2.5.4.9 Expand Groups - RMS Server

Goal: Obtain a user's group membership and email address.

Context of use: RMS servers use Group Expansion, part of the RMS: Server-to-Server Protocol, to determine group membership of authorized users across complex network directories.

Access policy on RMS-protected content can specify individual users as well as distribution groups. When a consumer contacts the RMS server for authorization to access protected content, the server might need to consult the directory to determine whether that user is a member of a group that is specified in the policy. If the group exists in a partition of the directory to which the RMS server does not have access, that RMS server needs to contact another server that does have appropriate permissions. This server-to-server communication can use either the **Group Expansion over SOAP** interface or the **Binary Group Expansion** interface.

The **Group Expansion over SOAP** interface exposes one request/response method: **IsPrincipalMemberOf**.

The **Binary Group Expansion** interface performs the same function as the **Group Expansion over SOAP** interface, however, it does so using a set of binary-formatted messages sent over HTTP. It exposes one request/response method: **IsPrincipalMemberOf**.

Direct actor: The direct actor of this use case is the RMS server that is the primary (first) server the client has contacted and that needs to authenticate the user.

Primary actor: The primary actor is the same as the direct actor.

Supporting actors: Any number of secondary RMS servers operating in other forests.

Note The terms "primary" and "secondary" are only used to distinguish the different actors in this use case. These terms are not meant to imply larger system relationships.

Stakeholders and interests:

- RMS client application, as described in section 2.5.1.
- RMS user, as described in section 2.5.1.
- Client computer, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.

Preconditions: The primary RMS server needs to have an SCP for Group Expansion in another forest. If the server does not meet this precondition, it can try to find a service location as described in section 2.5.4.10.

Minimal guarantees: The primary RMS server is able to determine the user's group membership, and retrieve the users' email address.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: The primary (or secondary) RMS server tries to determine the user's group membership, but the group membership is not found in the primary RMS server's domain.

Main Success Scenario:

1. The primary RMS server sends a request to one or more secondary RMS servers in other forest. For each secondary RMS server:

- If the secondary RMS server can find the user in its directory service, it responds to the primary RMS server with the appropriate group membership information and the user's email address.
- If the secondary RMS server cannot find the user is in its directory service, it also sends a Group Expansion request to one or more secondary RMS servers in other forests:

    1. The secondary RMS server starts at step 1 as if it were a primary RMS server.
    2. In the event of a successful response, the secondary RMS server responds back to the primary RMS server that sent the initial Service Location request.

2. The primary RMS server receives a response with the client's group membership and email address.

Note This is a recursive algorithm. Care should be taken to prevent recursions for the same request from being executed multiple times through each server or forest.

Extensions: None.

2.5.4.10 Find Service Locations for Group Expansion - RMS Server

Goal: Find the appropriate URLs for Group Expansion.

Context of use: RMS servers use Find Service Locations to find the Group Expansion interface on a remote server before making a cross-forest group expansion request.

Direct actor: The direct actor of this use case is the specific RMS server that has to authenticate a user.

Primary actor: The primary actor is the same as the direct actor.

Supporting actors: Any number of other RMS servers.

Stakeholders and interests: RMS server, as described in section 2.5.1

Preconditions: The specific RMS server already has a list of other RMS servers.

Minimal guarantees: The specific RMS server obtains a service location for Group Expansion on a server in a different forest.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: The specific RMS server has to perform group expansion, but does not have one or more service locations for the Group Expansion service.

Main success scenario:

1. The specific RMS server sends a Service Location request to one or more other RMS servers on its list, requesting the Group Expansion service.
2. The other RMS servers will each respond.
3. At least one of the responses includes a valid SCP for Group Expansion on another forest. This can be an SCP pointing to one of the other RMS servers that responded, or an SCP that is known to that server.
4. The specific RMS server continues by performing group expansion.

Extensions: None.

2.5.4.11 Decommission Server - ISV Application

Goal: Stop using the RMS system, while still allowing RMS-protected content to be used.

Context of use: If an organization were to decide to stop using RMS entirely and remove its deployment, it would need to remove RMS protection from content. One method is to have people with owner rights to each piece of content remove the protection. Realistically, however, it might not be possible to find these people since they might no longer belong to the organization in question. Another approach is to use the Decommissioning interface to extract the content key from a PL and return it so that it can then be used to decrypt the content. Because each protected document has a PL, and each PL has its own content key, this process is repeated for each protected document that needs to have its protection removed.

Note When servicing the request, the RMS server does not verify that the requestor is supposed to be granted access to the content as specified in the PL. Rather, the RMS server will return the content key to any requestor. As a result, the Decommissioning interface is disabled for normal operation by default.

Direct actor: RMS administrator.

Primary actor: The RMS server.

Supporting actors: Any other RMS servers in the system.

Stakeholders and interests:

- RMS administrator, as described in section 2.5.1.
- ISV application, as described in section 2.5.1.
- Client computer, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.

Preconditions: The RMS server has been bootstrapped into the system and has not yet been decommissioned.

Minimal guarantees: The content key is extracted from the PL. The protected content guarded by the PL can be decrypted using the content key.

Success guarantee: Content keys are extracted from all PLs. All protected content can be decrypted by using the corresponding content keys.

Trigger: RMS is being decommissioned at the enterprise level.

Main success scenario:

1. The RMS administrator enables decommissioning on the RMS server, so that the AcquireContentKey operation can be accessed.
2. The RMS administrator uses AcquireContentKey to request the content key for a PL in the RMS system, and decrypts the content guarded by that PL.
3. For the success guarantee, the RMS Administrator repeats step 2 for each PL in the RMS system.

Extensions: None.

2.5.4.12 Republishing Content - ISV Application

Goal: Create a new signed PL that has the same content key as an existing signed PL.

Context of use: This interface is useful when there is a need to alter the set of rights that were granted to users in the original PL without re-encrypting the content with a new content key.

Direct actor: RMS client.

Primary actor: The primary actor is the same as the direct actor.

Supporting actors: The RMS server.

Stakeholders and interests:

- RMS user, as described in section 2.5.1.
- ISV application, as described in section 2.5.1.
- Client computer, as described in section 2.5.1.
- RMS server, as described in section 2.5.1

Preconditions:

- The RMS server has been bootstrapped into the system.
- The RMS client is authorized to call the EditIssuanceLicense service.
- The RMS client has a signed PL that allows republishing.
- The RMS client has an unsigned PL containing the new set of rights.

Minimal guarantees: The ISV application has a signed PL for the content. This PL contains the updated set of rights for the content. The PL contains the location of the RMS server to retrieve a use license to consume the content. The PL allows the intended recipient to consume the content.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: An RMS user needs to replace the set of rights granted by a PL without changing the content key that protects the content.

Main success scenario:

1. The ISV application uses the EditIssuanceLicense operation to request an updated PL from the RMS server, submitting both the signed PL used to protect the content and an unsigned PL for the updated set of rights.
2. The RMS server adds the existing content key to the unsigned PL for the updated content, signs the PL, and returns it to the ISV application.

Extensions: None.

2.5.4.13 Perform Precertification - ISV Application

Goal: To acquire a license on behalf of a recipient user.

Context of use: When protected content is sent to recipients, each recipient acquires a use license that grants access to the content. The use license describes the usage policy for that user with that content and encrypts the content key to the user's public key.

Precertification optimizes the process. With precertification, the use license for a recipient could be generated in advance and made available with the content at the time the recipient attempted to access it. The use license could be requested on behalf of the recipient by either the sender or a server application that might be involved in delivering the content to the recipient. This use license would allow the recipient to access the content as soon as it was delivered without having to contact the RMS server, presuming that the recipient has already been bootstrapped.

Direct actor: ISV application.

Primary actor: The requestor (an RMS user).

Supporting actors: The RMS server and the recipient (an RMS user).

Stakeholders and interests:

- ISV application, as described in section 2.5.1.
- The requestor - an RMS user, as described in section 2.5.1.
- The recipient - an RMS user, as described in section 2.5.1.
- Client computer, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.

Preconditions:

- The client needs to be bootstrapped with the RMS server, per section 2.5.4.2.
- The client needs to be able to determine the location of the RMS server by the means specified in [MS-RMPR] section 3.8.3.2.

Minimal guarantees: The requestor has the public part of the recipient's RMS account certificate (RAC) and a use license (UL) allowing the recipient to use the protected content. The ISV application can then package the RAC and UL with the protected content, allowing the recipient to use the content as soon as it has been delivered.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: The requestor wants to ensure the recipient can use the content upon receipt.

Main success scenario:

1. The requestor uses an ISV application on the client machine to request the ability to grant rights to the recipient.
2. The ISV application contacts the RMS server, retrieving the public part of the recipient's RMS account certificate (RAC) by using the Precertification interface. The RMS server returns the public part of the RAC.
3. The ISV application requests a use license from the RMS server. The RMS server returns the use license.

Extensions: None.

2.5.4.14 Perform Prelicensing - ISV Application

Goal: To acquire a license on behalf of a set of recipients.

Context of use: The context of use includes the context of use for precertification (section 2.5.4.13), expanded to include the possible need to acquire use licenses for multiple recipients.

Prelicensing adds further optimization for client implementations by allowing the RMS server to act on the client's behalf. With the prelicensing operation, the ISV application sends a list of recipients to the RMS server instead of performing precertification separately on each recipient. The RMS server will perform precertification for each recipient, and return the associated public RACs and use licenses back to the client. This approach does not require ISV applications to have as many entry points into the RMS system as precertification.

Direct actor: RMS client.

Primary actor: The requestor (an RMS user).

Supporting actors: The RMS server and the recipient (an RMS user).

Stakeholders and interests:

- ISV application, as described in section 2.5.1.
- The requestor - an RMS user, as described in section 2.5.1.
- The recipients - a set of RMS users, as described in section 2.5.1.
- Client computer, as described in section 2.5.1.
- RMS server, as described in section 2.5.1.

Preconditions:

- The client needs to be bootstrapped with the RMS server, per section 2.5.4.2.
- The client needs to have a list of recipient's email addresses.
- The RMS client application needs to be able to determine the location of the RMS server by the means specified in [MS-RMPR] section 3.8.3.2.

Minimal guarantees: The requestor has a use license for each recipient.

Success guarantee: The success guarantee is the same as the minimal guarantee.

Trigger: The requestor wants to ensure the recipients can use the content upon receipt.

Main success scenario:

1. The requestor uses the ISV application on the client machine to request the ability to grant rights to a list of specific recipients.
2. The RMS client contacts the RMS server, indicating the set of recipients by using a list of email addresses.
3. For each recipient on the list, the RMS server performs precertification (see section 2.5.4.13) on behalf of the client. The public portion of each recipient's RAC is stored locally, along with the recipient's use license; this set of data will be the prelicensing result.

4. The RMS server returns the prelicensing result to the ISV application.

Extensions: None.

2.6 Versioning, Capability Negotiation, and Extensibility

The RMS protocols have client and server versions 1.0, 1.0 SP1, 1.0 SP2, and 2.0.

The RMS protocols support 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.

2.7 Error Handling

The RMS protocols are SOAP-based protocols that use HTTP 1.1 for transport. The protocols allow 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. For more information about the SOAP faults from the RMS server, see [MS-RMPR] section 3.1.4.5 and [MS-RMPRS] section 2.2.9.1.

Some common areas of failure are:

- Certificate or license format or arguments are invalid.
- Certificate or license has expired.
- Access is unauthorized.
- Email address is invalid or formatted incorrectly.
- Machine certificate (SPC) is not valid.
- A certificate or license has been tampered with (signature validation failed).
- VersionData element contains an invalid or unsupported version number.
- The user does not have rights in the PL.
- The certificate or license did not come from a trusted issuer.

The RMS protocols will not be able to function if there are failures on the services it depends on, such as network connectivity, availability of DNS, or availability of a directory service.

Note The **Binary Group Expansion** interface uses the HTTP transport ([RFC1945] and [RFC2616]) and does not generate SOAP faults.

2.8 Coherency Requirements

This system has no special coherency requirements.

2.9 Security

This section documents system-wide security issues that are not otherwise described in the technical documents (TDs) for the member protocols. It does not duplicate what is already in the member protocol TDs unless there is some unique aspect that applies to the system as a whole.

The system does not define any security requirements beyond those described in the specifications of the protocols supported by the system, as listed in section 2.2.

2.10 Additional Considerations

There are no additional considerations.

3 Examples

This section provides a series of examples illustrating the use of the RMS protocols. The examples are:

- Protecting content using offline publishing.
- Protecting content using online publishing.
- Consuming protected content.

Each example is described in detail in the following sections as a set of conceptual actions interchanged between the system and participants. These actions correspond to the interfaces described in section 2.1.2.2.

Each example, where applicable, assumes the client is well-informed of the appropriate SCP, obtained by using service location discovery, a directory service, or pre-existing local configuration data.

3.1 Example 1: Activating the RMS Servers

3.1.1 Activate the Server

Before participating in the RMS system, the RMS server enrolls as defined in [MS-RMPR] section 3.1.3. To enroll, the RMS server needs to have an SLC. Server enrollment requests can be made synchronously by the server directly contacting the RMS cloud service, or asynchronously by an RMS administrator exporting the enrollment request and contacting the RMS cloud service from another computer.<8> For more information on contacting the RMS cloud service, see [MS-RMPR] section 3.1.3.2.

3.1.2 Activate a Subordinate RMS Server

An RMS server can be deployed as a subordinate to another RMS server. A root RMS server grants the subordinate RMS server the right to perform only certain licensing tasks by issuing a subordinate SLC based its own SLC. For a subordinate RMS server, this process replaces the standard RMS server bootstrapping process defined in [MS-RMPR] section 3.1.3.

To enroll, the subordinate RMS server requests an SLC by sending a SubEnroll request ([MS-RMPRS] section 3.3.4.1) to the root RMS server. Then, to establish trust, the subordinate RMS server requests a limited licensor certificate from the root RMS server by using the GetLicensorCertificate operation ([MS-RMPRS] section 3.4.4.1).

3.2 Example 2: Using Offline Publishing to Protect Content

One of the most common tasks in RMS is publishing protected content using an RMS client application. This section describes the typical steps that can be completed for a user and computer that has not used RMS in the past to bootstrap the client, acquire a CLC, acquire templates and, finally, publish the content offline. Note that templates are an optional component of RMS but are included in this example.

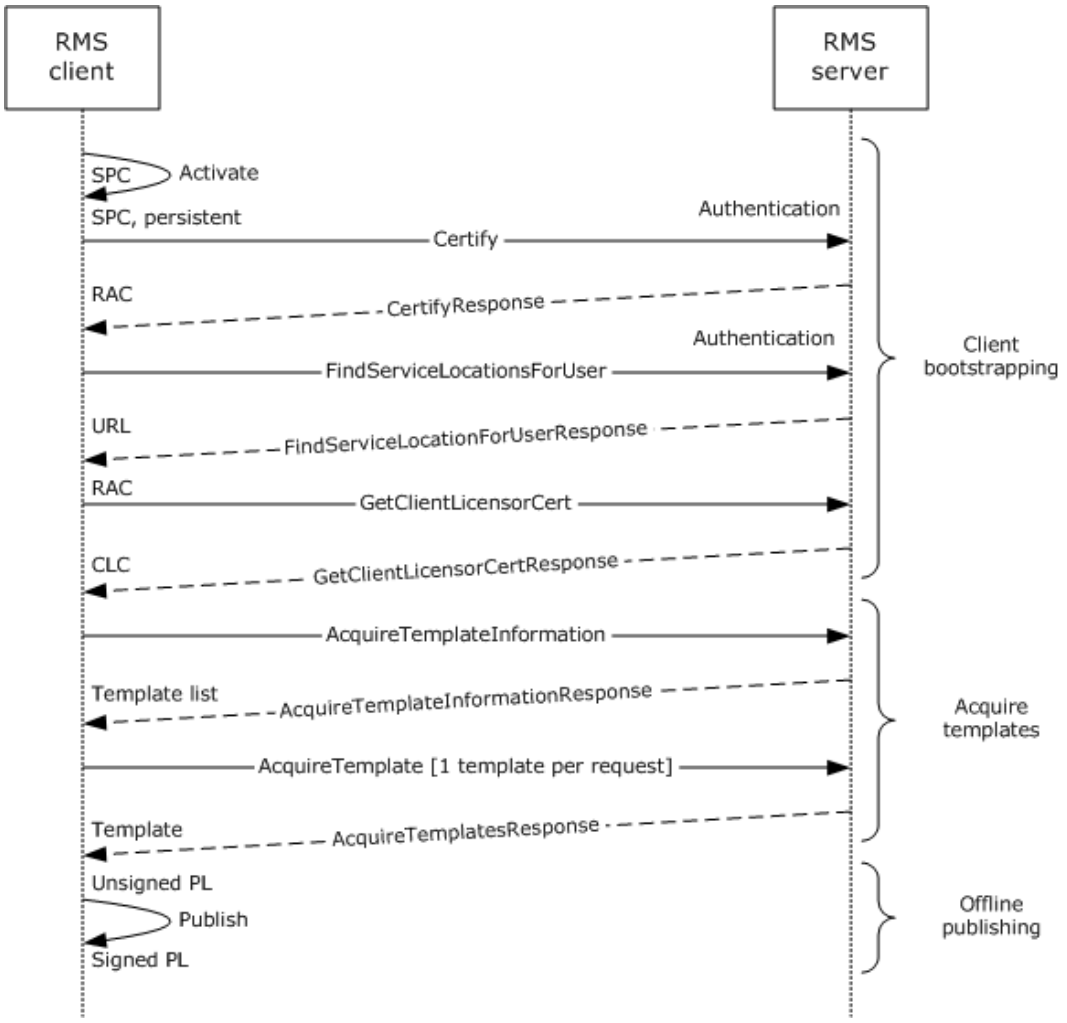

Figure 8: Message flow for protecting content using offline publishing

Note In figure 8, the Certify and FindServiceLocationsForUser calls are interchangeable. It is possible to call FindServiceLocationsForUser (to get the service location for GetClientLicensorCert) before calling Certify so long as both requests are completed before calling GetClientLicensorCert.

3.2.1 Client Bootstrapping

3.2.1.1 Activate the Computer

Before a computer or device participates in the RMS system it generates a security processor certificate (SPC). In RMS protocol versions 1.0 SP1 and later, clients self-activate, generating their own SPC.<9> More details on activation can be found in [MS-RMPR] section 3.2.4.1.

3.2.1.2 Find Service Locations

Depending on the deployment topology of the servers in the network, different servers can be used for different functions for a given user. The client makes a FindServiceLocationsForUser request to the RMS server, and in return receives the URL of the server to make subsequent calls to for client

bootstrapping. Details about the FindServiceLocationsForUser method can be found in [MS-RMPR] section 3.7.4.2.

3.2.1.3 Certify the User

After the computer is activated, the user is certified to participate in the RMS system. This is accomplished by binding an encryption key pair to both the user and the client computer by way of a RAC. The user has a RAC to access protected content or to publish protected content offline. Full details of client certification can be found in [MS-RMPR] section 3.3.4.1.

3.2.1.4 Acquire a CLC

To publish offline, the user has a signing key pair. The client licensor certificate (CLC) binds a signing key pair to the user through their RMS account certificate (RAC). The CLC private key is encrypted by the RAC public key, the SPC and RAC are required to decrypt the signing key. The client uses the GetClientLicensorCert method to acquire a CLC from that server. More information can be found in [MS-RMPR] section 3.5.4.2, which details the GetClientLicensorCert method.

3.2.2 Acquire Templates

An RMS client application can request rights policy templates from an RMS server. First the RMS client application 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 makes AcquireTemplate requests to the server to download the templates.<10> The client maintains a local store of templates and when subsequent AcquireTemplateInformation requests are made, the client compares the server list against the list in the local store.<11> The client makes add/delete/edit updates to the templates in the local store. Through this process, the client always keeps its templates in sync with the ones on the server.

3.2.3 Offline Publishing

Publishing offline is done entirely by the RMS client application and does not involve any interaction with the RMS server. When the client is used to protect content, it generates a PL that contains the usage policy and the content key, both of which are encrypted using the server's public key. The PL also contains a reference to a server that can be used to issue ULs from the PL.

During offline publishing, the usage policy and content key are encrypted using the server's public key from the ISSUER element 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. For more details on offline publishing see [MS-RMPR] section 1.3.5.

3.3 Example 3: Using Online Publishing to Protect Content

Content can be published by using online publishing, where the client does not need to maintain certificates locally and use the server to sign publishing licenses (PLs). In online publishing, the client computer is not required to be activated and the client user is not required to be bootstrapped. This example describes a typical scenario using online publishing, assuming rights policy templates are used in the publishing process.

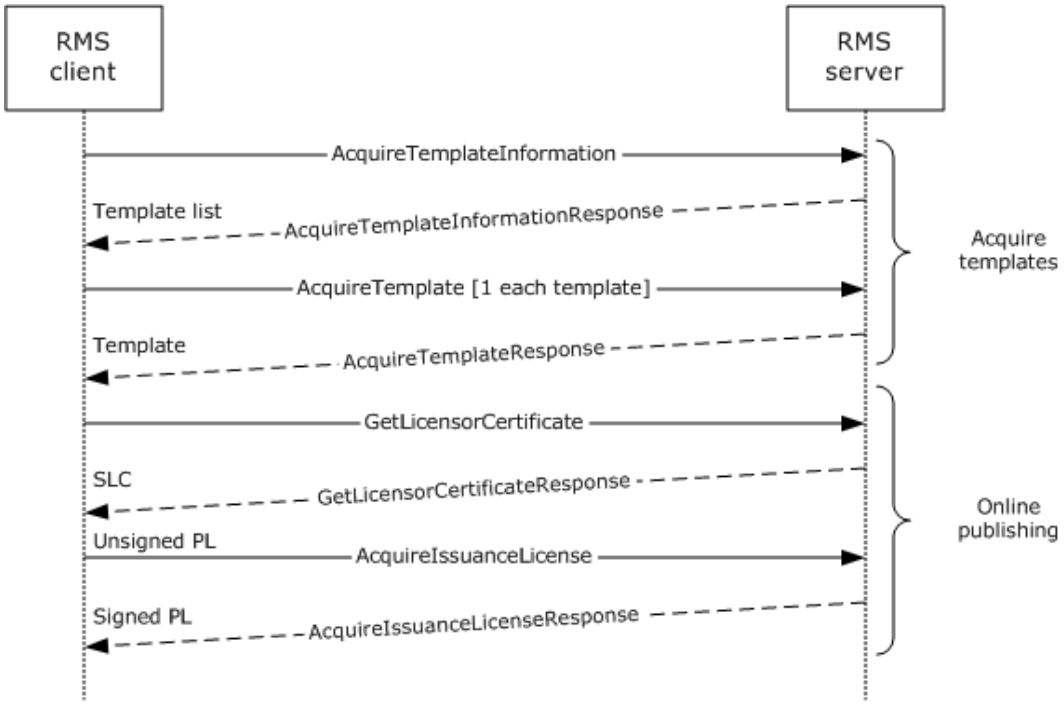

Figure 9: Message flow for protecting content using online publishing

3.3.1 Acquire Templates

An RMS client application can request rights policy templates from an RMS server. First the RMS client application 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 makes AcquireTemplate requests to the server to download the templates. The client maintains a local store of templates and when subsequent AcquireTemplateInformation requests are made, the client compares the server list against the list in the local store.<12> The client makes add/delete/edit updates to the templates in the local store. Through this process, the client always keeps its templates in sync with the ones on the server.

3.3.2 Online Publishing

3.3.2.1 Acquire the Server's Certificate

When publishing content, the RMS client application needs the public key from the server's SLC to encrypt the symmetric content key and usage rights in the PL. This way only the server can decrypt these elements, which it will do to generate use licenses. To get the server's SLC, the RMS client application sends a request to the GetLicensorCertificate method. No custom data is sent in this request, and when the server receives the request, it returns the SLC to the RMS client application.

3.3.2.2 Generate a Publishing License

After the RMS client application has the SLC and templates required for publishing, it generates a PL. The RMS server is not contacted during this step. The application includes the symmetric content key and use restrictions in the PL and encrypts them using the server's public key.

3.3.2.3 Sign the Publishing License

The final step in online publishing is to sign the PL. This is done by the RMS client application making an AcquireIssuanceLicense request to the RMS server. In the AcqurieIssuanceLicense request, the unsigned PL is sent to the server, the server signs it and returns the signed PL.

3.4 Example 4: Consuming Protected Content

When a user receives protected content they wish to consume, a series of steps have to happen to set them up in RMS and for the application to acquire a use license to open the protected content. This section describes the typical steps performed to consume content, assuming the computer and user have not used RMS in the past.

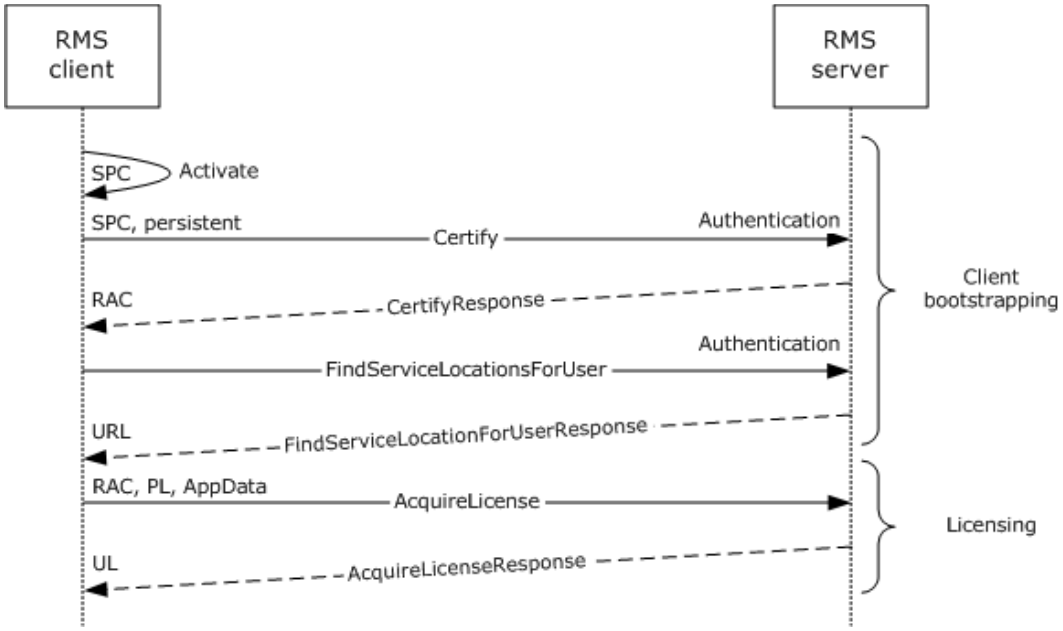

Figure 10: Message flow for consuming protected content

3.4.1 Client Bootstrapping

3.4.1.1 Activate the Computer

Before a computer or device can participate in the RMS system, it generates a security processor certificate (SPC). In RMS protocol versions 1.0 SP1 and later, clients self-activate, generating their own SPC.<13> More details on activation can be found at [MS-RMPR] section 3.2.4.1.

3.4.1.2 Certify the User

To access protected content, the user needs a RAC that corresponds to the user's account. After performing any necessary service discovery, the client uses the Certify request to acquire a RAC. The server issues an asymmetric encryption key pair and identifies the user account in the RMS system. The client needs a valid SPC before calling Certify. Full details of client certification can be found in [MS-RMPR] section 3.3.4.1.

3.4.2 Licensing

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.

To retrieve the UL, the RMS client application issues an AcquireLicense request. In an AcquireLicense request, the client sends the PL that came with the content along with the user's RAC and any application data to the RMS server. The server validates the RAC and PL and passes through any application data that is present. If validation succeeds and the user is granted access, the server generates a use license and returns it to the client. More details on AcquireLicense can be found in [MS-RMPR] section 3.4.4.1.

3.5 Example 5: Accessing the Server for Advanced Scenarios

3.5.1 Republishing Content

Republishing is performed when there is a need to alter the set of rights that were granted to users in the original PL, but without re-encrypting the content with a new content key.

To re-publish content, an ISV client application is used to request an updated PL from the RMS server. This is done using the EditIssuanceLicense operation ([MS-RMSI] section 3.4.4.1). This request includes both the signed PL used to protect the content and an unsigned PL for the updated set of rights. The RMS server then adds the existing content key to the unsigned PL for the updated content, signs the PL, and returns it to the ISV client application.

3.5.2 Perform Precertification

Precertification is used to create a use license (UL) for a recipient in advance, so that the UL can be provided with the content.

To precertify content, the requestor uses an ISV application on the client machine to request the ability to grant rights to the recipient. The ISV application contacts the RMS server, retrieving the public part of the recipient's RMS account certificate (RAC) by using the Precertify operation ([MS-RMSI] section 3.3.4.1). After receiving the public portion of the recipient's RAC, the ISV application gets the use license from the RMS server. This can be obtained either through implementation-specific connections to the RMS client or by calling the AcquireLicense operation directly ([MS-RMPR] section 3.4.4.1). The UL is then sent with the protected content, and allows the recipient to access the content as soon as it is delivered.

3.5.3 Perform Prelicensing

Like precertification, prelicensing is used to obtain use licenses (ULs) in advance. Prelicensing optimizes the process by allowing the RMS server to act on the client's behalf. Prelicensing does not require ISV applications to obtain use licenses by means other than the ISV extensions; instead, use licenses are acquired by the server and returned as part of the prelicensing operation.

To prelicense content, the requestor uses an ISV application on the client machine to request the ability to grant rights to one or more recipients. The ISV application sends a list of recipients to the RMS server by using the AcquirePreLicense operation ([MS-RMSI] section 3.5.4.1). The set of recipients is indicated with a list of email addresses. The RMS server performs precertification for each recipient and returns the associated use licenses to the ISV client application.

3.5.4 Decommission Server

Decommissioning is used to extract the content key from a PL for use with decrypting the content. The content can then be accessed outside of the RMS system, or after the RMS system has been deactivated at the enterprise level.

To initiate decommissioning, the RMS administrator enables the Decommissioning interface on the RMS server. This interface has to be enabled so that the AcquireContentKey operation can be accessed. (The interface is disabled by default, to prevent unauthorized access; the RMS server does not verify the requestor's access in the PL.) For each piece of protected content that is being decommissioned out of the RMS system, the RMS administrator uses the AcquireContentKey operation ([MS-RMSI] section 3.2.4.1) to request the content key for the protected content's PL. Then, the protected content guarded by that PL is decrypted by using that content's key. This process is repeated for each protected document that needs to have its protection removed.

4 Microsoft Implementations

The information in this specification is applicable to the following versions of Windows:

- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system

Exceptions, if any, are noted in the following section.

4.1 Product Behavior

<1> Section 2.1.5: Support for SHA256 encryption of message signature hashes was introduced in Windows Server 2012. Implementations prior to Windows Server 2012 only support SHA-1. Support for SHA256 can be added to Windows Server 2008 R2 by installing a QFE [MSKB2627272].

<2> Section 2.2: Windows products that support ISV extensions include Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2.

<3> Section 2.5.1: All versions of Microsoft's 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.

<4> Section 2.5.4.1: All versions of Microsoft's RMS server earlier than version 2 contacted the Microsoft enrollment cloud 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.

<5> Section 2.5.4.2: Windows RMS clients will search Active Directory for the SCP unless one of the following registry keys is present.

- "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDRM\ServiceLocation\Activation" can be used to specify the location of the certification service, using the format: http(s)://servername/_wmcs/certification.

- "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDRM\ServiceLocation\EnterprisePublishing" can be used to specify the location of the licensing service, using the format: http(s)://servername/_wmcs/licensing.

In addition, applications can specify an alternate service URL when invoking Windows APIs that would normally search Active Directory for the SCP.

Windows RMS servers will search Active Directory for the SCP unless the GICURL value of one of the following registry keys contains the location of the certification service, http(s)://servername/_wmcs/certification.

- For RMS 1.0 SP2 or earlier, the registry key is "HKEY_LOCAL_MACHINE\Software\Microsoft\DRMS\1.0".
- For Windows Server 2008, the registry key is "HKEY_LOCAL_MACHINE\Software\Microsoft\DRMS\2.0".
- For Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, the registry key is "HKEY_LOCAL_MACHINE\Software\Microsoft\DRMS".

<6> Section 2.5.4.3: All versions of Microsoft's RMS server earlier than version 2 used sub-enrollment to sign the SLC into the hierarchy. RMS version 2 neither calls nor exposes the sub-enrollment interface. 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.

<7> Section 2.5.4.5: RMS 2.0 provides an interface for clients to retrieve rights policy templates. RMS versions prior to 2.0 do not provide this interface.

<8> Section 3.1.1: All versions of Microsoft's RMS server earlier than version 2 contacted the Microsoft enrollment cloud 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.

<9> Section 3.2.1.1: Support for the RMS Client version 1.0 has ended, and the cloud activation service is no longer available for activation requests. Activate requests from RMS 1.0 clients will still be made to the RMS server, but activation requests from the RMS server to the cloud service will fail. This failure will result in the server returning a failure to the RMS client.

RMS: Client-to-Server protocol versions 1.0 SP1, 1.0 SP2, and 2.0 use self-activation. Self-activation continues to function as expected.

<10> Section 3.2.2: Windows implementations transmit licenses in batches of 25.

<11> Section 3.2.2: In Windows implementations, the client does not maintain the list of templates as part of client and server operations. A Scheduled Tasks job is provided that can be enabled and is run separately to maintain the local store of templates.

<12> Section 3.3.1: In Windows implementations, the client does not maintain the list of templates as part of client and server operations. A Scheduled Tasks job is provided that can be enabled and is run separately to maintain the local store of templates.

<13> Section 3.4.1.1: Support for the RMS client version 1.0 has ended, and the Cloud Activation Service is no longer available for activation requests. Activate requests from RMS 1.0 clients will still be made to the RMS server, but activation requests from the RMS server to the cloud service will fail. This failure will result in the server returning a failure to the RMS client.

RMS: Client-to-Server protocol versions 1.0 SP1, 1.0 SP2, and 2.0 use self-activation. Self-activation continues to function as expected.

5 Change Tracking

This section identifies changes that were made to the [MS-RMSOD] protocol document between the January 2013 and August 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.3 References	Removed reference for [MS-DISO].	N	Content updated.
2.1.5 Relevant Standards	Removed the reference to [MS-DISO].	N	Content updated.
2.3.2.3 Directory Services	Removed the reference to [MS-DISO].	N	Content updated.
4 Microsoft Implementations	Modified this section to include references to Windows 8.1 operating system and Windows Server 2012 R2 operating system.	Y	Content updated.

6 Index

F

G

H

I

M

O

P

R

S

T

U

V