[MS-GRVSPCM]:
Client to Management Server Groove SOAP Protocol Specification

---

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
07/13/2009	1.02	Major	Revised and edited the technical content
08/28/2009	1.03	Editorial	Revised and edited the technical content
11/06/2009	1.04	Editorial	Revised and edited the technical content
02/19/2010	2.0	Minor	Updated the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.05	Minor	Clarified the meaning of the technical content.
09/27/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	3.0	Major	Significantly changed the technical content.
06/10/2011	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	3.1	Minor	Clarified the meaning of the technical content.
04/11/2012	3.1	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Client to Management Server SOAP Protocol. The protocol is a framework of services used to:

- Obtain identity and **policy (2)** information.
- Upload account backups, **audit logs**, client usage statistics.
- Enable users to do client account password resets.

  This protocol is a request/response message exchange protocol built on top of a custom implementation of Groove SOAP.

The protocol consists of 24 interfaces, 6 service categories:

1. Bootstrap services
   - Auto account code configuration
   - Auto activation
   - Create account
   - Domain enrollment
   - Domain migration status
   - Enrollment
   - Failures (device management)
   - Key activation
2. Management services
   - GMS Configuration
   - Account heartbeat
   - Managed object install
   - Managed object status
   - Statistics package
3. Backup services
   - Account store
4. Directory services
   - Identity publish
   - Contact search
   - Contact fetch
5. Password management services

- Manual passphrase reset request
- Manual passphrase reset request status
- Automatic password reset

6. Audit services
   - Audit log upload
   - Audit log upload query
   - File upload
   - File upload query

The protocol depends on the proper use of these interfaces. This document specifies the proper use of all six categories of services.

Additionally, this document specifies the following:

- How messages are encapsulated on the wire, common data types, and certificate requirements in section 2: Messages
- Protocol details, including client and server details, in section 3: Protocol Details
- Protocol examples in section 4: Protocol Examples
- Protocol security considerations in section 5: Security

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

certificate
certification authority (CA)
Distinguished Encoding Rules (DER)
domain member (member machine)
encryption
Group Policy Object (GPO)
GUID
Hash-based Message Authentication Code (HMAC)
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
little-endian
Message Authentication Code (MAC)
object identifier (OID)
private key
public key
public key infrastructure (PKI)
secret key
Secure Sockets Layer (SSL)

symmetric key
Unicode
UTF-8

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

account
account configuration code
Advanced Encryption Standard (AES)
audit log
device
device URL
ElGamal encryption
identity URL
Modified Alleged Rivest Cipher 4 (MARC4) algorithm
policy
presence server
session
shared space
SOAP fault
telespace
Uniform Resource Identifier (URI)
Uniform Resource Locator (URL)
vCard
XML namespace
XML schema
XML schema definition (XSD)

The following terms are specific to this document:

domain certificate: An X.509 certificate (2) that is associated with a management domain, as described in [X509]. It contains the public key that is used to help secure registration transactions between protocol clients and protocol servers.

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

1.2 References

References to Microsoft Open Specifications documentation 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.

1.2.1 Normative References

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

[CRYPTO] Menezes, A., Vanstone, S., and Oorschot, P., "Handbook of Applied Cryptography", 1997, http://www.cacr.math.uwaterloo.ca/hac/

[MS-GRVSPMR] Microsoft Corporation, "Management Server to Relay Server Groove SOAP Protocol Specification".

[PKCS1] RSA Laboratories, "PKCS #1: RSA Cryptography Standard", PKCS #1, Version 2.1, June 2002, http://www.rsa.com/rsalabs/node.asp?id=2125

[PKCS5] RSA Laboratories, "PKCS #5: Password-Based Cryptography Standard", PKCS #5, Version 2.0, March 1999, http://www.rsa.com/rsalabs/node.asp?id=2127

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

[RFC2279] Yergeau, F., "UTF-8, A Transformation Format of ISO10646", RFC 2279, January 1998, http://www.ietf.org/rfc/rfc2279.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

[RFC3174] Eastlake III, D., and Jones, P., "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001, http://www.ietf.org/rfc/rfc3174.txt

[RFC3280] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002, http://www.ietf.org/rfc/rfc3280.txt

[RFC4634] Eastlake III, D., and Hansen, T., "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006, http://www.ietf.org/rfc/rfc4634.txt

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, http://www.ietf.org/rfc/rfc4648.txt

[SCHNEIER] Schneier, B., "Applied Cryptography, Second Edition", John Wiley and Sons, 1996, ISBN: 0471117099.

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

[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/

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA] World Wide Web Consortium, "XML Schema", September 2005, http://www.w3.org/2001/XMLSchema

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

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

[XMLSchemaInstance] W3C, "XML Schema Instance", 1999, http://www.w3.org/1999/XMLSchema-instance

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

1.2.2 Informative References

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

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

[RFC2426] Dawson, F., and Howes, T., "vCard MIME Directory Profile", RFC 2426, September 1998, http://www.rfc-editor.org/rfc/rfc2426.txt

[XMLSchema1999] W3C, "DTD for XML Schemas: Part 1: Structures", 1999, http://www.w3.org/1999/XMLSchema

1.3 Protocol Overview (Synopsis)

The Client to Management Server Groove SOAP Protocol provides support for managing clients through identity and **device** management and policy distribution. The Client to Management Server Groove SOAP Protocol is a request/response message exchange protocol based on Groove SOAP that uses HTTP 1.1 as its transport. The following diagram provides an overview of this protocol.

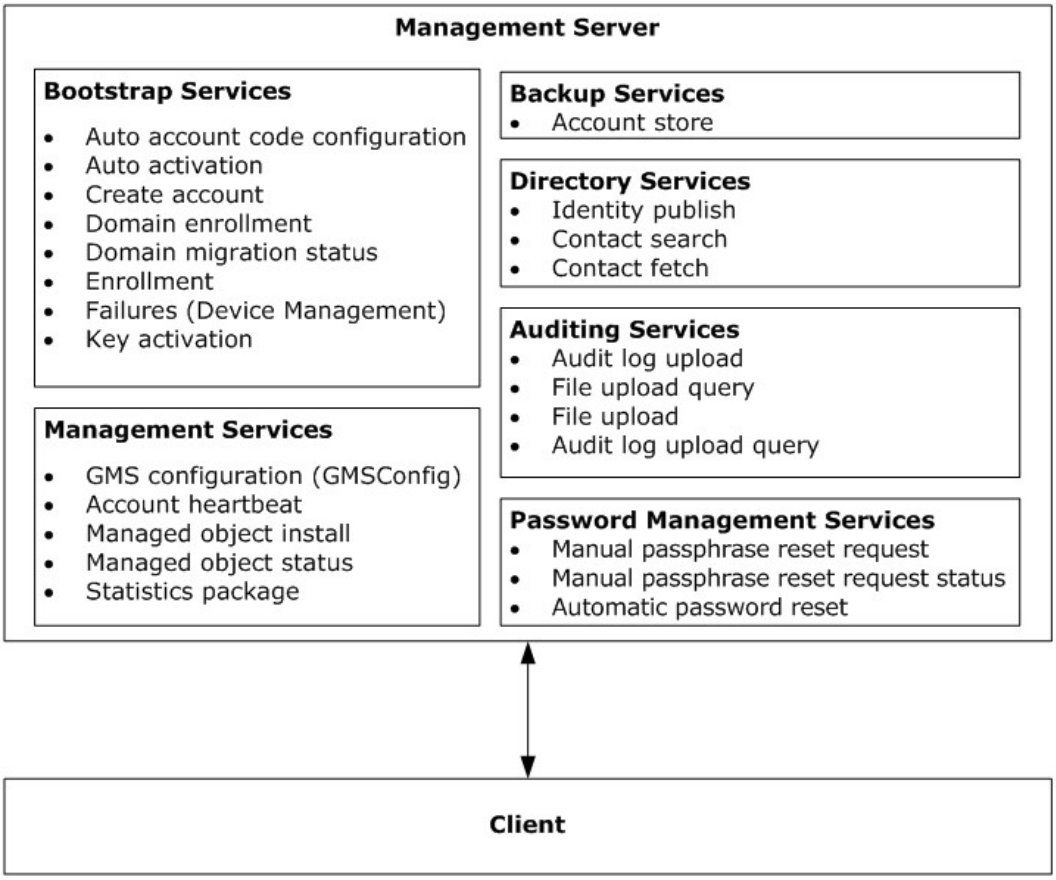

Figure 1: Management services

The Client to Management Server Groove SOAP Protocol involves two active entities: the client and management server (the server).

An administrator on the server first creates a domain and populates the domain with members. The server creates a unique **account configuration code** for every new member created. The client uses either the configuration code or the operating system login name to bind a client member to a server member.

On successful binding, the client contacts the server for

- Periodic polling of member and policy update
- Periodic backup of **account** data
- Member contact search
- Account password reset
- Reporting on client usage
- Uploading client audit log data

The role of the server in the protocol is to manage members, distribute security and policy information, sign identity contacts, enable users to reset account passwords, provide storage for account backups, store audit logs and to collect client usage statistics.

The protocol consists of a single server endpoint. The server endpoint is responsible for servicing the request.

The protocol is a session-less protocol. The following diagram illustrates the message exchange.

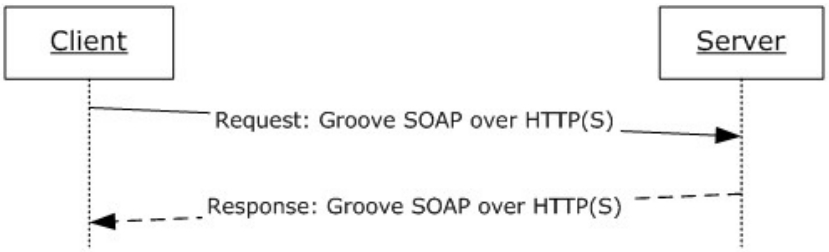

Figure 2: Message interaction between protocol client and protocol server

1.3.1 Service Interfaces

1.3.1.1 GMS Configuration

The client sends an empty request to http://<server>/GMSConfig server URL. The server responds with response header containing server URLs for posting authenticated and non-authenticated transactions.

1.3.1.2 Account Heartbeat

The client sends an account heartbeat message to enable the server to mark the account and the device as active.

1.3.1.3 Account Store

The client sends an account store message to back up an account to the server.

1.3.1.4 Audit Log Upload

The client sends an audit log upload message for uploading client audit logs to the management server.

1.3.1.5 Audit Log Upload Query

The client sends this message for fetching the last device audit log sequence number from server for opportunistic purging of audit log data.

1.3.1.6 Auto Account Code Configuration

On bootstrap, a newly installed client with an auto account configuration **Group Policy Object (GPO)** policy defined will use the logged in user's login credentials to initiate an auto account code configuration message to the server. The server on successful association between the client user and a server member based on user's login name will respond by sending either:

- A copy of backed up account data for the logged in user, which is used by the client to restore the user account or
- A set of managed objects, each managed object contains a member or policy details.

1.3.1.7 Auto-Activation

On bootstrap, a newly installed client with an auto-activation policy defined uses the logged-on user's logon credentials to initiate an auto-activation message to the server. The server creates an association between the client user and management server member based on user name. On successful association, server responds by sending the account configuration code for the member. The client on successful auto-activation response uses the account configuration code to send a key activation message.

1.3.1.8 Automatic Password Reset

The client uses automatic password reset message to enable a user to self-service client password reset requests. The server, on successful automatic password reset request, sends an e-mail containing a temporary password to the user.

1.3.1.9 Contact Fetch

The client uses the contact fetch message to fetch one or more contacts from the server's contact directory.

1.3.1.10 Contact Search

The client uses the contact search message to search the server's contact directory.

1.3.1.11 Create Account

The client uses create the account message to register an account/management domain pair with the server. The purpose of the message is to deliver a **secret key** to the server. The client and the server use the secret key for encrypting and integrity protecting subsequent messages between the client and server for that account domain pair.

1.3.1.12 Domain Enrollment

The domain enrollment message completes the account configuration process. The client sends a domain enrollment message to retrieve the identity's contact signed by the server.

1.3.1.13 Domain Migration Status

The client sends this message to inform the server on the identity's migration status from the current domain to a new one.

1.3.1.14 Enrollment

The client sends an enrollment message in response to an enrollment required fault code from the server. The client sends an enrollment message to retrieve the identity's contact signed by the server.

1.3.1.15 Failures (Device Management)

The client uses the failures message to report on policy based device management failures to the server.

1.3.1.16 File Upload

The client uses the file upload message to upload audited client files to the server.

1.3.1.17 File Upload Query

The client uses the file upload query message to query the server if another device has uploaded the audited client file / revision.

1.3.1.18 Identity Publish

The identity publish message is used by client to publish a member's contact to the server's contact directory.

1.3.1.19 Key Activation

The client uses key activation message to associate a client identity with a member on the server. On successful association, the server responds by sending member data and associated policies to the client in a series of managed objects. Each managed object contains a member or a policy detail.

1.3.1.20 Managed Object Install

The client sends a managed object install message to the server on successful installation of a managed object.

1.3.1.21 Managed Object Status

The client uses managed object status message on periodic basis to test for member or policy updates from the server.

1.3.1.22 Manual Passphrase Reset Request

The client uses the manual passphrase reset request message to request a manual passphrase reset from the server.

1.3.1.23 Manual Passphrase Reset Request Status

The client uses the manual passphrase reset request status message to test the status of a manual passphrase reset request. The client sends this message until a server administrator approves or disapproves the passphrase reset request or the user cancels the passphrase reset request.

1.3.1.24 Statistics Package

The client uses the statistics package message to upload client usage and other statistics to the server.

1.3.2 Protocol Security

The protocol has built-in security. All messages to the server except for the auto account code configuration message are encrypted and integrity-protected to ensure the confidentiality and integrity of the message. The auto account code configuration message uses HTTPS for securing its payload content.

The account configuration messages, key activation message, and account creation message from the bootstrap services enable the security.

The key activation message uses an account configuration code and a **Uniform Resource Locator (URL)** that the client obtains from the server using an out-of-band means (e-mail is one possibility). The client encrypts the request with a key derived from the account configuration code. The server, upon receiving the request, identifies a member associated with the configuration code, and sends back the member's identity template along with other policy information. The identity template specifies which management domain the identity belongs to and the domain's **certificate (1)**.

The auto account code configuration message allows a user to configure an account with the management server without the need for an account configuration code. The client uses HTTPS to post an auto account code configuration message to an authenticated server endpoint. The server, upon receiving the request, identifies a member associated with the login name, and sends back the member's identity template along with other policy information. The identity template specifies which management domain the identity belongs to and the domain's certificate.

Once the client has obtained the **domain certificate**, the client uses the create account message to establish the shared secret key between the client and the server. To do so, the client generates a 192-bit secret key, encrypts it with the domain's **encryption public key** and signs the message with its own signature **private key**. The client sends the message along with its own public key information to the server. The server verifies the signature and decrypts the message to get the shared key.

On successful registration of the secret key, all subsequent requests are secured with this key. The responses are not always secured. For simple responses with just return codes, the responses are not secured. For responses with payload, they are secured unless HTTPS is used.

The auto activation message is used by a managed device that has already obtained a registry setting for the device management domain (which contains among other things, the device domain's certificate, the device domain **GUID**, the server URL). This message behaves slightly differently from the auto account code configuration message. The client first uses the create account message

to create a client device account on the server and to establish a shared key between the client device and the server. The client then sends the auto activation message, encrypted with the shared device key to an authenticated server endpoint. Upon receiving the request, the server decrypts the request, finds the matching account with the login name, and returns the account configuration code, again encrypted with the shared device key. The client then uses the key activation request to get the identity template, which contains the identity's management domain certificate. Finally it creates a shared account key by using the create account message with the identity's domain certificate.

1.4 Relationship to Other Protocols

The protocol uses secure Groove SOAP for formatting requests and responses. It transmits these messages using the **Hypertext Transfer Protocol (HTTP)** or the **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)** protocols. Groove SOAP is the wire format used for messaging, and HTTP or HTTPS are the underlying transport protocols. A management domain's public key or an account configuration code or an agreed upon secret key is used for encrypting and signing and integrity protecting the payload content.

The following diagram illustrates the transport stack that the protocol uses:

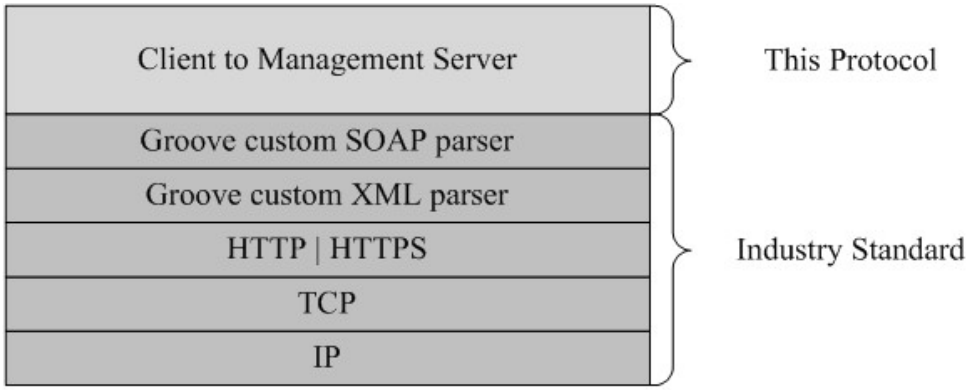

Figure 3: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

To use the protocol defined by this specification, the client has to be able to establish a connection to a management server over TCP/IP using the well-known HTTP or HTTPS ports.

The client associates a client identity with a management server member via one of the bootstrap services provided by the protocol before other services can be used.

This protocol operates against a server that is identified by a URL that is known by protocol clients (section 3.2.5.3.1).

1.6 Applicability Statement

None.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The protocol MUST support Groove SOAP over HTTP (as specified in [RFC2616]) over TCP/IP. The protocol MUST use HTTPS for auto account configuration message.

2.2 Message Syntax

This section defines the grammar of messages used by the protocol using **XML schema definition (XSD)** and template. For more information about XSD, see [XMLSCHEMA1] and [XMLSCHEMA2]. XPath is used to address the elements and attributes in **XML schemas**. For more information about XPATH, see [XPATH].

2.2.1 Namespaces

This specification uses the following **XML namespace Uniform Resource Identifier (URI)** values (see [XMLNS] for information about XML namespaces).

Prefix	Namespace URI	Reference
SOAP-ENC	http://schemas.xmlsoap.org/soap/encoding/	[SOAP1.1]
SOAP-ENV	http://schemas.xmlsoap.org/soap/envelope/	[SOAP1.1]
xs	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA]
xsd	http://www.w3.org/1999/XMLSchema	[XMLSchema1999]
xsi	http://www.w3.org/1999/XMLSchema-instance	[XMLSchemaInstance]
g	urn:groove.net	

2.2.2 Common Schema

The elements and data types described in this section are used in multiple services.

All certificates, when specified as attribute values, are always **Distinguished Encoding Rules (DER)** encoded.

The following specify the protocol message schemas.

Request Message Schemas

```
<xs:schema xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="Envelope">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Body">
   <xs:complexType>
```

```
    <xs:sequence>
     <xs:choice>
     <xs:element ref="AccountHeartbeat"/>
     <xs:element ref="AccountStore"/>
     <xs:element ref="AuditLogUpload"/>
     <xs:element ref="AuditLogUploadQuery"/>
     <xs:element ref="AutoAccountCodeConfiguration"/>
     <xs:element ref="AutoActivation"/>
     <xs:element ref="AutomaticPasswordReset"/>
     <xs:element ref="ContactFetch"/>
     <xs:element ref="ContactSearch"/>
     <xs:element ref="CreateAccount"/>
     <xs:element ref="DomainEnrollment"/>
     <xs:element ref="DomainMigrationStatus"/>
     <xs:element ref="Enrollment"/>
     <xs:element ref="Failures"/>
     <xs:element ref="FileUpload"/>
     <xs:element ref="FileUploadQuery"/>
     <xs:element ref="IdentityPublish"/>
     <xs:element ref="KeyActivation"/>
     <xs:element ref="ManagedObjectInstall"/>
     <xs:element ref="ManagedObjectStatus"/>
     <xs:element ref="PassphraseResetRequest"/>
     <xs:element ref="PassphraseResetStatus"/>
     <xs:element ref="StatisticsPackage"/>
     </xs:choice>
    </xs:sequence>
    </xs:complexType>
   </xs:element>
   </xs:sequence>
   <xs:attribute ref="SOAP-ENV:encodingStyle" use="required"/>
  </xs:complexType>
  </xs:element>
  <xs:attribute name="encodingStyle" type="xs:string"/>
 </xs:schema>
```

The referenced child elements of the **Body** element are specified in the following schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="http://www.w3.org/1999/XMLSchema-instance"/>
 <xs:element name="AccountHeartbeat" type="ServiceRequestType1"/>
 <xs:element name="AccountStore" type="ServiceRequestType1"/>
 <xs:element name="AuditLogUpload" type="ServiceRequestType1"/>
 <xs:element name="AuditLogUploadQuery" type="ServiceRequestType1"/>
 <xs:element name="AutoAccountCodeConfiguration" type="ServiceRequestType3"/>
 <xs:element name="AutoActivation" type="ServiceRequestType1"/>
 <xs:element name="AutomaticPasswordReset" type="ServiceRequestType1"/>
 <xs:element name="ContactFetch" type="ServiceRequestType1"/>
 <xs:element name="ContactSearch" type="ServiceRequestType1"/>
 <xs:element name="CreateAccount" type="ServiceRequestType2"/>
 <xs:element name="DomainEnrollment" type="ServiceRequestType3"/>
 <xs:element name="DomainMigrationStatus" type="ServiceRequestType1"/>
 <xs:element name="Enrollment" type="ServiceRequestType1"/>
 <xs:element name="Failures" type="ServiceRequestType1"/>
 <xs:element name="FileUpload" type="ServiceRequestType1"/>
 <xs:element name="FileUploadQuery" type="ServiceRequestType1"/>
 <xs:element name="IdentityPublish" type="ServiceRequestType1"/>
```

```
 <xs:element name="KeyActivation" type="ServiceRequestType3"/>
 <xs:element name="ManagedObjectInstall" type="ServiceRequestType1"/>
 <xs:element name="ManagedObjectStatus" type="ServiceRequestType1"/>
 <xs:element name="PassphraseResetRequest" type="ServiceRequestType1"/>
 <xs:element name="PassphraseResetStatus" type="ServiceRequestType1"/>
 <xs:element name="StatisticsPackage" type="ServiceRequestType1"/>

 <xs:complexType name="ServiceRequestType1">
 <xs:sequence>
  <xs:element name="Payload" type="ContentPayloadType"/>
  <xs:element name="Version" type="NumericType"/>
  <xs:element name="LastBroadcastProcessed" type="NumericType"/>
  <xs:element name="MessageSequenceNumber" type="NumericType"/>
 </xs:sequence>
 </xs:complexType>

 <xs:complexType name="ServiceRequestType2">
 <xs:sequence>
  <xs:element name="Payload" type="ContentPayloadType"/>
  <xs:element name="Version" type="NumericType"/>
  <xs:element name="LastBroadcastProcessed" type="NumericType"/>
 </xs:sequence>
 </xs:complexType>

 <xs:complexType name="ServiceRequestType3">
 <xs:sequence>
  <xs:element name="Payload" type="AttributePayloadType"/>
  <xs:element name="Version" type="NumericType"/>
 </xs:sequence>
 </xs:complexType>

 <xs:complexType name="AttributePayloadType">
 <xs:attribute name="data" type="xs:base64Binary" use="required"/>
 <xs:attribute
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
ref="xsi:type"
use="required"
fixed="binary"/>
 </xs:complexType>

 <xs:complexType name="ContentPayloadType">
 <xs:simpleContent>
  <xs:extension base="xs:base64Binary">
  <xs:attribute
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
ref="xsi:type"
use="required"
fixed="base64"/>
  </xs:extension>
 </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="NumericType">
 <xs:simpleContent>
  <xs:extension base="xsd:int">
  <xs:attribute
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
ref="xsi:type"
use="required"
```

```
fixed="xsd:int"/>
  </xs:extension>
 </xs:simpleContent>
 </xs:complexType>
</xs:schema>
```

The referenced "xsi:type" is specified in the following schema:

```
<xs:schema xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:attribute name="type" type="xs:string"/>
</xs:schema>
```

Response Message Schemas

```
<xs:schema xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="Envelope">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Body">
   <xs:complexType>
   <xs:sequence>
    <xs:choice>
    <xs:element ref="AccountHeartbeatResponse"/>
    <xs:element ref="AccountStoreResponse"/>
    <xs:element ref="AuditLogUploadResponse"/>
    <xs:element ref="AuditLogUploadQueryResponse"/>
    <xs:element ref="AutoAccountCodeConfigurationResponse"/>
    <xs:element ref="AutoActivationResponse"/>
    <xs:element ref="AutomaticPasswordResetResponse"/>
    <xs:element ref="CreateAccountResponse"/>
    <xs:element ref="ContactFetchResponse"/>
    <xs:element ref="ContactSearchResponse"/>
    <xs:element ref="DomainEnrollmentResponse"/>
    <xs:element ref="DomainMigrationStatusResponse"/>
    <xs:element ref="EnrollmentResponse"/>
    <xs:element ref="FailuresResponse"/>
    <xs:element ref="FileUploadResponse"/>
    <xs:element ref="FileUploadQueryResponse"/>
    <xs:element ref="IdentityPublishReponse"/>
    <xs:element ref="KeyActivationResponse"/>
    <xs:element ref="ManagedObjectInstallResponse"/>
    <xs:element ref="ManagedObjectStatusResponse"/>
    <xs:element ref="PassphraseResetRequestResponse"/>
    <xs:element ref="PassphraseResetStatusResponse"/>
    <xs:element ref="StatisticsPackageResponse"/>
    </xs:choice>
   </xs:sequence>
   </xs:complexType>
```

```
  </xs:element>
  </xs:sequence>
  <xs:attribute ref="SOAP-ENV:encodingStyle" use="required"/>
 </xs:complexType>
 </xs:element>
 <xs:attribute name="encodingStyle" type="xs:string"/>
</xs:schema>
```

The referenced child elements of the **Body** element are specified in the following schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="http://www.w3.org/1999/XMLSchema-instance"/>
 <xs:element name="AccountHeartbeatResponse" type="ServiceResponseType1"/>
 <xs:element name="AccountStoreResponse" type="ServiceResponseType1"/>
 <xs:element name="AuditLogUploadResponse" type="ServiceResponseType2"/>
 <xs:element name="AuditLogUploadQueryResponse"
             type="ServiceResponseType2"/>
 <xs:element name="AutoAccountCodeConfigurationResponse"
             type="ServiceResponseType2"/>
 <xs:element name="AutoActivationResponse" type="ServiceResponseType2"/>
 <xs:element name="AutomaticPasswordResetResponse"
             type="ServiceResponseType2"/>
 <xs:element name="CreateAccountResponse" type="ServiceResponseType1"/>
 <xs:element name="ContactFetchResponse" type="ServiceResponseType2"/>
 <xs:element name="ContactSearchResponse" type="ServiceResponseType2"/>
 <xs:element name="DomainEnrollmentResponse" type="ServiceResponseType2"/>
 <xs:element name="DomainMigrationStatusResponse"
           type="ServiceResponseType1"/>
 <xs:element name="EnrollmentResponse" type="ServiceResponseType1"/>
 <xs:element name="FailuresResponse" type="ServiceResponseType1"/>
 <xs:element name="FileUploadResponse" type="ServiceResponseType2"/>
 <xs:element name="FileUploadQueryResponse" type="ServiceResponseType2"/>
 <xs:element name="IdentityPublishReponse" type="ServiceResponseType1"/>
 <xs:element name="KeyActivationResponse" type="ServiceResponseType2"/>
 <xs:element name="ManagedObjectInstallResponse"
          type="ServiceResponseType1"/>
 <xs:element name="ManagedObjectStatusResponse"
             type="ServiceResponseType3"/>
 <xs:element name="PassphraseResetRequestResponse"
             type="ServiceResponseType1"/>
 <xs:element name="PassphraseResetStatusResponse"
             type="ServiceResponseType2"/>
 <xs:element name="StatisticsPackageResponse" type="ServiceResponseType1"/>
 <xs:complexType name="ServiceResponseType1">
 <xs:sequence>
  <xs:element name="ReturnCode" type="NumericType"/>
 </xs:sequence>
 </xs:complexType>

 <xs:complexType name="ServiceResponseType2">
 <xs:sequence>
  <xs:element name="ReturnCode" type="NumericType"/>
  <xs:element name="Payload" type="AttributePayloadType"/>
 </xs:sequence>
 </xs:complexType>

 <xs:complexType name="ServiceResponseType3">
```

```
 <xs:sequence>
  <xs:element name="ReturnCode" type="NumericType"/>
  <xs:element name="ManagedObjects" minOccurs="0"
              type="AttributePayloadType"/>
 </xs:sequence>
 </xs:complexType>

 <xs:complexType name="AttributePayloadType">
 <xs:attribute name="data" type="xs:base64Binary" use="required"/>
 <xs:attribute
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
ref="xsi:type"
use="required"
fixed="binary"/>
 </xs:complexType>

 <xs:complexType name="NumericType">
 <xs:simpleContent>
  <xs:extension base="xsd:int">
  <xs:attribute
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
ref="xsi:type"
use="required"
fixed="xsd:int"/>
  </xs:extension>
 </xs:simpleContent>
 </xs:complexType>
</xs:schema>
```

The referenced "xsi:type" is specified in the following schema:

```
<xs:schema xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:attribute name="type" type="xs:string"/>
</xs:schema>
```

The following specifies the service fault response message schema:

```
<xs:schema xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="Envelope">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Body">
   <xs:complexType>
   <xs:sequence>
    <xs:element name="Fault" type="ServiceFaultResponseType"/>
   </xs:sequence>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
  <xs:attribute ref="SOAP-ENV:encodingStyle" use="required"/>
 </xs:complexType>
```

```
</xs:element>

<xs:complexType name="ServiceFaultResponseType">
<xs:sequence>
 <xs:element name="faultCode" type="xs:int"/>
 <xs:element name="faultString" type="xs:string"/>
</xs:sequence>
</xs:complexType>

 <xs:attribute name="encodingStyle" type="xs:string"/>
</xs:schema>
```

2.2.2.1 Common Elements

The following table summarizes the set of common XSD element definitions specified by this protocol. XSD element definitions that are only used in a single message are specified in section 2.2.3 of this document.

Element	Description
Event	Message event element
Fault	Service fault response element
fragment	Fragment element
KeyActivation	Key activation element
LastBroadcastProcessed	Last broadcast processed element
ManagementDomain	Management domain element
MessageSequenceNumber	Message sequence number element
Payload	Payload data element
PayloadWrapper	Payload wrapper element
ReturnCode	Return code element
ReturnPayloadWrapper	Return payload wrapper element
Version	Version element
SE	Secured element

2.2.2.1.1 Event Element

The **Event** element contains message event information.

```
<xs:element name="Event" type="EventType"/>
```

The **EventType** is specified in section 2.2.2.2.7.

2.2.2.1.2 Fault Element

The **Fault** element is used in a service fault response.

```
<xs:element name="Fault" type="ServiceFaultResponseType"/>
```

The **ServiceFaultResponseType** is specified in section 2.2.2.2.15.

2.2.2.1.3 fragment Element

This **fragment** element contains data fragment of the **Event** element specified in section 2.2.2.1.1.

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="Event"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

2.2.2.1.4 KeyActivation Element

The **KeyActivation** element contains key activation information.

```
<xs:element name="KeyActivation" type="ActivationType"/>
```

The **ActivationType** is specified in section 2.2.2.2.1.

2.2.2.1.5 LastBroadcastProcessed Element

The **LastBroadcastProcessed** element is used to indicate the last server command executed on the client.

```
<xs:element name="LastBroadcastProcessed" type="NumericType"/>
```

The **NumericType** is specified in 2.2.2.2.11.

2.2.2.1.6 ManagementDomain Element

The **ManagementDomain** element contains managed domain information.

```
<xs:element name="ManagementDomain" type="ManagementDomainType"
            xmlns:g="urn:groove.net"/>
```

The **ManagementDomainType** is specified in section 2.2.2.2.9.

2.2.2.1.7 MessageSequenceNumber Element

The **MessageSequenceNumber** element MUST be ignored by the server.

```
<xs:element name="MessageSequenceNumber" type="NumericType"/>
```

The **NumericType** is specified in section 2.2.2.2.11.

2.2.2.1.8 Payload Element

The **Payload** element is the payload for a service.

For an attribute payload message, the **Payload** element is specified as:

```
<xs:element name="Payload" type="AttributePayloadType"/>
```

For a content payload message, the **Payload** element is specified as:

```
<xs:element name="Payload" type="ContentPayloadType"/>
```

The **AttributePayloadType** and the **ContentPayloadType** types are specified in sections 2.2.2.2.2 and 2.2.2.2.5.

2.2.2.1.9 PayloadWrapper Element

The **PayloadWrapper** element contains payload data.

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="PayloadWrapper">
 <xs:complexType>
  <xs:sequence>
  <xs:element xmlns:g="urn:groove.net" ref="g:SE"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The referenced "g:SE" element is specified in the following schema. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="SE" type="KeyWrapperType"/>
</xs:schema>
```

The **KeyWrapperType** type is specified in section 2.2.2.2.8.

2.2.2.1.10 ReturnCode Element

The **ReturnCode** element contains a service status code in a service response.

```
<xs:element name="ReturnCode" type="NumericType"/>
```

The **NumericType** type is specified in section 2.2.2.2.11.

2.2.2.1.11 ReturnPayloadWrapper Element

The **ReturnPayloadWrapper** element contains data in a service response.

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="ReturnPayloadWrapper">
 <xs:complexType>
  <xs:sequence>
  <xs:element xmlns:g="urn:groove.net" ref="g:SE"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The referenced "g:SE" element is specified in section 2.2.2.1.13. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

2.2.2.1.12 Version Element

The **Version** element contains service version information.

```
<xs:element name="Version" type="NumericType"/>
```

The **NumericType** type is specified in section 2.2.2.2.11.

2.2.2.1.13 SE Element

The **SE** contains secured data.

```
<xs:element name="SE" type="SEType" xmlns:g="urn:groove.net"/>
```

The **SEType** is specified in section 2.2.2.2.22.

2.2.2.2 Common Types

The following table summarizes the set of common XSD type definitions specified by this protocol. XSD type definitions that are only used in a single message are specified in section 2.2.3 of this document.

Type	Description
ActivationType	Account activation type

Type	Description
AttributePayloadType	Service payload type that uses the "data" attribute as payload
BooleanType	A restricted xs:boolean type
ContactType	Contact data type
ContentPayloadType	Service payload type that uses element content as payload
CSecurityType	Contact security data type
EventType	Message event type
KeyWrapperType	Security key data wrapper type
ManagementDomainType	Management domain type
ManagedObjectType	Managed object type
NumericType	Extended from xsd:int type
ObjectHeaderType	Managed object header type
ObjectSignatureType	Managed object signature type
PKIPolicyType	**public key infrastructure (PKI)** policy type
ServiceFaultResponseType	Service fault response type
ServiceRequestType1	A service request type
ServiceRequestType2	A service request type
ServiceRequestType3	A service request type
ServiceResponseType1	A service response type
ServiceResponseType2	A service response type
ServiceResponseType3	A service response type
SEType	Secure element type

2.2.2.2.1 ActivationType

The **ActivationType** contains account activation data.

```
<xs:complexType name="ActivationType">
 <xs:sequence>
 <xs:element xmlns:g="urn:groove.net" ref="g:ManagementDomain"/>
 <xs:element name="ManagedObjects">
  <xs:complexType>
  <xs:sequence>
   <xs:element name="ManagedObject" type="ManagedObjectType"
       maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="Count" type="xs:int" use="required"/>
  </xs:complexType>
 </xs:element>
```

```
 </xs:sequence>
 <xs:attribute name="ActivationKey" type="xs:string" use="required"/>
 <xs:attribute name="ServerURL" type="xs:string" use="required"/>
</xs:complexType>
```

The referenced "g:ManagementDomain" element is specified in section 2.2.2.1.6. The ManagementDomain element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net". The **ManagedObjectType** is specified in section 2.2.2.2.10.

The following table describes the elements and attributes of the type:

XPath	Description
/ActivationType/@ActivationKey	Activation key
/ActivationType/@ServerURL	Activation server URL
/ActivationType/ManagementDomain	Management domain element
/ActivationType/ManagementObjects	Domain managed objects element
/ActivationType/ManagementObjects/@Count	Managed object count
/ActivationType/ManagementObjects/fragment/ManagedObject	Managed object element

2.2.2.2.2 AttributePayloadType

The **AttributePayloadType** contains service data using the "data" attribute.

```
<xs:complexType name="AttributePayloadType">
 <xs:sequence>
 <xs:attribute name="data" type="xs:base64Binary" use="required"/>
 <xs:attribute ref="xsi:type" use="required" fixed="binary"/>
 </xs:sequence>
</xs:complexType>
```

The following table describes the attributes of the type:

XPath	Description
/AttributePayloadType/@data	Service response payload data.
/AttributePayloadType/@type	Payload data type. Its value is "binary".

Payload data are service specific.

2.2.2.2.3 BooleanType

The **BooleanType** is specified as:

```
<xs:simpleType name="BooleanType">
 <xs:restriction base="xs:boolean">
  <xs:pattern value="0|1"/>
 </xs:restriction>
```

```
</xs:simpleType>
```

The value 1 represents true; and the value 0 represents false.

2.2.2.2.4 ContactType

The **ContactType** is specified as:

```
<xs:complexType name="ContactType">
 <xs:sequence>
 <xs:element name="vCard">
  <xs:complexType>
  <xs:attribute name="Data" type="xs:base64Binary" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="ClientDevices" type="ClientDevicesType"/>
 <xs:element name="RelayDevices" type="RelayDevicesType"/>
 <xs:element name="CSecurity" type="CSecurityType"/>
 </xs:sequence>
 <xs:attribute name="Flags" type="xs:int" use="required"/>
 <xs:attribute name="SeqNum" type="xs:int" use="required"/>
 <xs:attribute name="URL" type="xs:string" use="required"/>
 <xs:attribute name="Version" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="ClientDevicesType">
 <xs:sequence>
 <xs:element name="Device">
  <xs:complexType>
  <xs:sequence>
   <xs:element name="PresenceDevices">
   <xs:complexType>
    <xs:sequence>
    <xs:element name="Device">
     <xs:complexType>
     <xs:attribute name="URL" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    </xs:sequence>
   </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="PlatformVersion" type="xs:string" use="required"/>
  <xs:attribute name="URL" type="xs:string" use="required"/>
  </xs:complexType>
 </xs:element>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="RelayDevicesType">
 <xs:sequence>
 <xs:element name="Device">
  <xs:complexType>
  <xs:attribute name="URL" type="xs:string" use="required"/>
  </xs:complexType>
 </xs:element>
 </xs:sequence>
```

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

The **CSecurityType** is specified in section 2.2.2.2.6.

The following table describes the payload XML elements and attributes:

XPath	Description
/ContactType/@SeqNum	Sequence number
/ContactType/@URL	Identity URL
/ContactType/@Version	Contact version
/ContactType/vCard	**vCard** element
/ContactType/vCard/@Data	This is a vCard serialized with UTF-8 encoding as defined in [RFC2426].
/ContactType/ClientDevices	Client devices data element
/ContactType/ClientDevices/Device	Client device data element
/ContactType/ClientDevices/Device/@PlatformVersion	Client device platform version
/ContactType/ClientDevices/Device/@URL	Client device URL
/ContactType/ClientDevices/Device/PresenceDevices	**Presence server** devices element
/ContactType/ClientDevices/Device/PresenceDevices/Device	Presence server device element
/ContactType/ClientDevices/Device/PresenceDevices/Device/@URL	Presence server device URL
/ContactType/RelayDevices	Relay devices element
/ContactType/RelayDevices/Device	Relay device element
/ContactType/RelayDevices/Device/@URL	Relay device URL
/ContactType/CSecurity	Contact security data element

2.2.2.2.5 ContentPayloadType

The **ContentPayloadType** contains service data using element content.

```
<xs:complexType name="ContentPayloadType">
 <xs:simpleContent>
 <xs:extension base="xs:base64Binary">
  <xs:attribute ref="xsi:type" use="required" fixed="base64"/>
 </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

The following table describes the attribute of the type:

XPath	Description
/ContentPayloadType/@type	The content data type. Its value is "base64".

Payload data are service specific.

2.2.2.2.6 CSecurityType

The **CSecurityType** contains security data.

```
<xs:complexType name="CSecurityType">
 <xs:sequence>
 <xs:element name="Algos">
  <xs:complexType>
  <xs:attribute name="EncAlgo" type="xs:string" use="required"/>
  <xs:attribute name="EncKeyAlgo" type="xs:string" use="required" fixed="DH"/>
  <xs:attribute name="SigAlgo" type="xs:string" use="required" fixed="RSA"/>
  <xs:attribute name="SigKeyAlgo" type="xs:string" use="required" fixed="RSA"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="Settings">
  <xs:complexType>
  <xs:attribute name="CipherAlgo" type="xs:string" use="required"
      fixed="MARC4-BM"/>
  <xs:attribute name="DigestAlgo" type="xs:string" use="required"
      fixed="SHA1"/>
  <xs:attribute name="Encrypted" type="BooleanType" use="required"/>
  <xs:attribute name="SKeyAlgo" type="xs:string" use="required"
      fixed="ARC4"/>
  </xs:complexType>
 </xs:element>
 </xs:sequence>
 <xs:attribute name="EPubKey" type="xs:base64Binary" use="required"/>
 <xs:attribute name="SPubKey" type="xs:base64Binary" use="required"/>
 <xs:attribute name="SelfSignature" type="xs:base64Binary" use="required"/>
</xs:complexType>
```

The following table describes the elements and attributes:

XPath	Description
/CSecurityType/@EPubKey	Encryption public key, DER encoded
/CSecurityType/@SPubKey	Signature public key, DER encoded
/CSecurityType/@SelfSignature	Self-signature, an encrypted element. The client can set this to any valid base64Binary value. The server MUST ignore the value of the SelfSignature.
/CSecurityType/Algos	Algorithm element
/CSecurityType/Algos/@EncAlgo	Encryption algorithm. The value MUST be "RSA" or "ELGAMAL".
/CSecurityType/Algos/@EncKeyAlgo	Encryption key algorithm. The value MUST be "DH".
/CSecurityType/Algos/@SigAlgo	Signature algorithm. The value MUST be "RSA".

XPath	Description
/CSecurityType/Algos/@SigKeyAlgo	Signature key algorithm. The value MUST be "RSA".
/CSecurityType/Settings	Settings element
/CSecurityType/Settings/@CipherAlgo	Cipher algorithm. The value MUST be "MARC4-BM".
/CSecurityType/Settings/@DigestAlgo	Digest algorithm. The value MUST be "SHA1".
/CSecurityType/Settings/@Encrypted	The value MUST be 1.
/CSecurityType/Settings/@SKeyAlgo	Secret key algorithm. The value MUST be "ARC4".

2.2.2.2.7 EventType

The **EventType** is specified as:

```
<xs:complexType name="EventType">
 <xs:sequence>
 <xs:element xmlns:g="urn:groove.net" ref="g:SE"/>
 </xs:sequence>
 <xs:attribute name="DeviceGUID" type="xs:string"/>
 <xs:attribute name="DomainGUID" type="xs:string" use="required"/>
 <xs:attribute name="EventID" type="xs:int"/>
 <xs:attribute name="GUID" type="xs:string" use="required"/>
 <xs:attribute name="GrooveVersion" type="xs:string" use="required"/>
 <xs:attribute name="HighPriority" type="BooleanType"/>
 <xs:attribute name="IdentityURL" type="xs:string" use="required"/>
 <xs:attribute name="IsDeviceAccount" type="BooleanType" use="required"/>
 <xs:attribute name="UniqueID" type="xs:string"/>
 <xs:attribute name="UserDeviceGuid" type="xs:string" use="required"/>
 <xs:attribute name="UserDeviceName" type="xs:string" use="required"/>
 <xs:attribute name="_EventID" type="xs:int" use="required"/>
 <xs:attribute name="created" type="xs:int" use="required"/>
</xs:complexType>
```

The "g:SE" element is specified in section 2.2.2.1.13. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net". The **BooleanType** is specified in section 2.2.2.2.3.

The following table describes the elements and attributes:

XPath	Description
/EventType/@DeviceGuid	The GUID of the device account
/EventType/@DomainGuid	Domain GUID
/EventType/@EventID	Event identifier (same as the "/EventType/@_EventID")
/EventType/@GUID	Event account GUID
/EventType/@GrooveVersion	Client version
/EventType/@HighPriority	Ignored by the server

XPath	Description
/EventType/@IdentityURL	**Identity URL**
/EventType/@IsDeviceAccount	A Boolean value MUST be true for a device account.
/EventType/@UniqueID	Ignored by the server
/EventType/@UserDeviceGuid	Device GUID
/EventType/@UserDeviceName	Client host name
/EventType/@_EventID	Event identifier
/EventType/@created	Message creation timestamp in Coordinated Universal Time (UTC).

2.2.2.2.8 KeyWrapperType

The **KeyWrapperType** contains key data.

```
<xs:complexType name="KeyWrapperType">
 <xs:extension base="SEType">
 <xs:attribute name="KeyID" type="xs:base64Binary" use="required"/>
 </xs:extension>
</xs:complexType>
```

The **SEType** is specified in section 2.2.2.2.22.

The following table describes the attribute of the type:

XPath	Description
/KeyWrapperType/@KeyID	Key identifier

2.2.2.2.9 ManagementDomainType

The **ManagementDomainType** contains management domain data.

```
<xs:complexType name="ManagementDomainType">
 <xs:attribute name="Certificate" type="xs:base64Binary" use="required"/>
 <xs:attribute name="DisplayName" type="xs:string" use="required"/>
 <xs:attribute name="Name" type="xs:string" use="required"/>
 <xs:attribute name="ReportingInterval" type="xs:int"
     use="required"/>
 <xs:attribute name="ReportingPolicy" type="xs:string" use="required"/>
 <xs:attribute name="ServerURL" type="xs:string" use="required"/>
</xs:complexType>
```

The following table describes the elements and attributes of the type:

XPath	Description
/ManagementDomainType/@Certificate	Domain certificate.

XPath	Description
/ManagementDomainType/@DisplayName	Domain display name
/ManagementDomainType/@Name	Domain GUID
/ManagementDomainType/@ReportingInterval	Domain reporting interval. The value MUST be 60.
/ManagementDomainType/@ReportingPolicy	Domain reporting policy. The value MUST be "Management"
/ManagementDomainType/@ServerURL	Domain server URL

2.2.2.2.10 ManagedObjectType

The **ManagedObjectType** contains managed object data.

```
<xs:complexType name="ManagedObjectType">
  <xs:attribute name="Active" type="BooleanType" use="required"/>
  <xs:attribute name="GUID" type="xs:string" use="required"/>
  <xs:attribute name="Name" type="xs:string" use="required"/>
  <xs:attribute name="Object" type="xs:base64Binary" use="required"/>
 </xs:complexType>
```

The following table describes the attributes of the type:

XPath	Description
/fragment/ManagedObject/@Active	A Boolean value that MUST be true to indicate an active status.
/fragment/ManagedObject/@GUID	The object GUID
/fragment/ManagedObject/@Name	The object name
/fragment/ManagedObject/@Object	The object data

The following specifies the /fragment/ManagedObject/@Object data schemas.

2.2.2.2.10.1 AccountServicesPolicy Object

The following specifies the **AccountServicesPolicy** object schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="fragment">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Header" type="ObjectHeaderType"/>
   <xs:element name="Body">
    <xs:complexType>
    <xs:sequence>
```

```
      <xs:element name="Policy">
      <xs:complexType>
       <xs:attribute name="Flags" type="xs:int"/>
      </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name="ComponentResourceURL" type="xs:string"
                 use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Signatures">
     <xs:complexType>
     <xs:sequence>
      <xs:element name="Signature" type="ObjectSignatureType"/>
     </xs:sequence>
     </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:attribute name="Version" type="xs:string" use="required"/>
   </xs:complexType>
   </xs:element>
  </xs:sequence>
  </xs:complexType>
 </xs:element>

 </xs:schema>
```

The **ObjectHeaderType** and the **ObjectSignatureType** are specified in sections 2.2.2.2.12 and 2.2.2.2.13.

The following table describes the elements and attributes:

XPath	Description
/fragment	Object data fragment element
/fragment/ManagedObject/@Version	The value MUST be "0,0,0,0"
/fragment/ManagedObject/Header	Object header element
/fragment/ManagedObject/Body	Object body element
/fragment/ManagedObject/Body/@ComponentResourceURL	The value MUST be "http://components.groove.net/Groove/Components/Root.osd?Package=net.groove.Groove.SystemComponents.GrooveAccountMgr_DLL&Version=0&Factory=AccountServicesPolicy"
/fragment/ManagedObject/Body/Policy	Policy element
/fragment/ManagedObject/Body/Policy/@Flags	The value MUST be one the following values or a value produced by using a bitwise OR operation on two or more of the following values: 0x00: No restrictions 0x01: Device cannot create another account

XPath	Description
	0x02: Device cannot import an account 0x04: Device can only use managed identities from device's management domain.<1>
/fragment/ManagedObject/Signatures	Signatures element
/fragment/ManagedObject/Signatures/Signature	Signature element

2.2.2.2.10.2 ComponentUpdatePolicy Object

The following specifies the **ComponentUpdatePolicy** object schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="fragment">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Header" type="ObjectHeaderType"/>
   <xs:element name="Body">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="ComponentUpdatePolicy">
     <xs:complexType>
      <xs:sequence>
      <xs:element name="PolicyItem" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
       <xs:sequence>
        <xs:element name="ComponetPackageNameList" minOccurs="0"
                  maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
         <xs:element name="ComponentPackageNameItem" minOccurs="0"
                   maxOccurs="unbounded">
          <xs:complexType>
           <xs:attribute name="Flags" type="xs:int"/>
           <xs:attribute name="Name" type="xs:string"/>
           <xs:attribute name="Policy" type="xs:string"/>
           <xs:attribute name="Version" type="xs:string"/>
           <xs:attribute name="VersionComparison" type="xs:string"/>
          </xs:complexType>
         </xs:element>
         </xs:sequence>
        </xs:complexType>
        </xs:element>
       </xs:sequence>
       <xs:attribute name="DisplayName" type="xs:string"/>
       <xs:attribute name="Fingerprint" type="xs:string"/>
       <xs:attribute name="Policy" type="xs:string"/>
       <xs:attribute name="Type" type="xs:string"/>
```

```
        </xs:complexType>
       </xs:element>
       </xs:sequence>
       <xs:attribute name="Default" type="xs:string"/>
       <xs:attribute name="Policyversion" type="xs:string"/>
       <xs:attribute name="SelfSigned" type="xs:string"/>
      </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name="ComponentResourceURL" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Signatures">
     <xs:complexType>
     <xs:sequence>
      <xs:element name="Signature" type="ObjectSignatureType"/>
     </xs:sequence>
     </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:attribute name="Version" type="xs:string" use="required"/>
   </xs:complexType>
   </xs:element>
  </xs:sequence>
  </xs:complexType>
 </xs:element>
 </xs:schema>
```

The **ObjectHeaderType** and the **ObjectSignatureType** are specified in sections 2.2.2.2.12 and 2.2.2.2.13.

The following table describes the elements and attributes:

XPath	Description
/fragment	Fragment element
/fragment/ManagedObject/@Version	The attribute value MUST be "0,0,0,0"
/fragment/ManagedObject/Header	Object header element
/fragment/ManagedObject/Body	Object body element
/fragment/ManagedObject/Body/@ComponentResourceURL	The value MUST be "http://components.groove.net/Groove/Components/Root.osd?Package=net.groove.Groove.SystemComponents.GrooveAccountMgr_DLL&Version=0&Factory=ComponentUpdatePolicy"
/fragment/ManagedObject/Body/ComponentUpdatePolicy	Component update policy element
/fragment/ManagedObject/Body/ComponentUpdatePolicy/@Default	Policy default. The value MUST be one of the following values: "Allow", "Deny", or "Local".
/fragment/ManagedObject/Body/ComponentUpdatePolicy/@Policyversion	Policy version
/fragment/ManagedObject/Body/ComponentUp	The value MUST be one of the following values: "Allow" or

XPath	Description
datePolicy/@SelfSigned	"Deny".
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem	Policy item element. Omit if no policy items.
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/@DisplayName	Display name element
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/@Fingerprint	Fingerprint
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/@Policy	The value MUST be one of the following values: "Allow", "Deny", or "Local".
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/@Type	The value MUST be one of the following values: "Signer" or "Use".
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/ComponentPackageNameList	Component package name list element
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/ComponentPackageNameList/ComponentPackageNameItem	Component package name item element
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/ComponentPackageNameList/ComponentPackageNameItem/@Flags	The value MUST be one of the following values or a value produced by using a bitwise OR operation on two or more of the following values: 0x00: do not apply any of the following options 0x01: no "missing tool" UI when usage is disallowed 0x02: delete tools data on disk when disallowed
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/ComponentPackageNameList/ComponentPackageNameItem/@Name	Package name
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/ComponentPackageNameList/ComponentPackageNameItem/@Policy	The value MUST be one of the following values: "Allow", "Deny", or "Local".
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/ComponentPackageNameList/ComponentPackageNameItem/@Version	Policy version
/fragment/ManagedObject/Body/ComponentUpdatePolicy/PolicyItem/ComponentPackageNameList/ComponentPackageNameItem/@VersionComparison	The value MUST be one of the following values: ">" >=", "&lt", "&lt=", or"=". Where "&gt" represents the ">" character and "&lt" represents the "<" character.
/fragment/ManagedObject/Signatures	Signatures element
/fragment/ManagedObject/Signatures/Signature	Signature element

2.2.2.2.10.3 DataRecoveryPolicy Object

The following specifies the **DataRecoveryPolicy** object schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="fragment">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Header" type="ObjectHeaderType"/>
   <xs:element name="Body">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="Policy">
     <xs:complexType>
      <xs:sequence>
      <xs:element name="Reset" minOccurs="0">
       <xs:complexType>
       <xs:attribute name="Text" type="xs:string"/>
       </xs:complexType>
      </xs:element>
      </xs:sequence>
      <xs:attribute name="Certificate" type="xs:base64Binary"/>
      <xs:attribute name="Flags" type="xs:int"/>
      <xs:attribute name="RecoveryType" type="xs:string"/>
     </xs:complexType>
     </xs:element>
     </xs:sequence>
     <xs:attribute name="ComponentResourceURL" type="xs:string" use="required"/>
    </xs:complexType>
    </xs:element>
    <xs:element name="Signatures">
    <xs:complexType>
     <xs:sequence>
     <xs:element name="Signature" type="ObjectSignatureType"/>
     </xs:sequence>
    </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="Version" type="xs:string" use="required"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
 </xs:complexType>
</xs:element>

</xs:schema>
```

The **ObjectHeaderType** and the **ObjectSignatureType** are specified in sections 2.2.2.2.12 and 2.2.2.2.13.

The following table describes the elements and attributes:

XPath	Description
/fragment	Object data fragment element

XPath	Description
/fragment/ManagedObject/@Version	The attribute MUST be "0,0,0,0".
/fragment/ManagedObject/Header	Object header element
/fragment/ManagedObject/Body	Object body element
/fragment/ManagedObject/Body/@ComponentResourceURL	The value MUST be "http://components.groove.net/Groove/Components/Root.osd?Package=net.groove.Groove.SystemComponents.GrooveAccountMgr_DLL&Version=0&Factory=DataRecoveryPolicy".
/fragment/ManagedObject/Body/Policy	Policy element
/fragment/ManagedObject/Body/Policy/@Certificate	MUST contain a domain data recovery certificate.
/fragment/ManagedObject/Body/Policy/@RecoveryType	MUST be set to "Full" for full data recovery, or "None" for no data recovery.
/fragment/ManagedObject/Body/Policy/@Flags	The attribute MUST be set to 1 to enable automatic password reset.
/fragment/ManagedObject/Body/Policy/Reset	Manual password reset instructions element. Omit if no reset instructions.
/fragment/ManagedObject/Body/Policy/Reset/@Text	Manual password reset instructions.
/fragment/ManagedObject/Signatures	Contains signature data
/fragment/ManagedObject/Signatures/Signature	Signature data

2.2.2.2.10.4 DevicePolicy Object

The following specifies the **DevicePolicy** object schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="fragment">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Header" type="ObjectHeaderType"/>
   <xs:element name="Body">
    <xs:complexType>
```

```
<xs:sequence>
 <xs:element name="Policy">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Bandwidth" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
   <xs:attribute name="DisplayMask" type="xs:string"/>
   <xs:attribute name="Value" type="xs:int"/>
   <xs:attribute name="Unit" type="xs:int"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="Audit" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
   <xs:sequence>
    <xs:element name="Client" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:attribute name="Flags" type="xs:int"/>
    </xs:complexType>
    </xs:element>
   </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="WebServices" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
   <xs:attribute name="IsDirectEnabled" type="xs:int"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="PKI" type="PKIPolicyType" minOccurs="0"
            maxOccurs="unbounded">
  <xs:element name="AuditServer" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
   <xs:sequence>
    <xs:element name="AuditItems" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
     <xs:element name="AuditItem" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
      <xs:attribute name="ComponentURL" type="xs:string"/>
      </xs:complexType>
     </xs:element>
     </xs:sequence>
    </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="Flags" type="xs:int"/>
   <xs:attribute name="ServerURL" type="xs:string"/>
   <xs:attribute name="UploadFrequency" type="xs:int"/>
   <xs:attribute name="UploadFrequency2" type="xs:int"/>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
  <xs:attribute name="Flags" type="xs:int"/>
 </xs:complexType>
 </xs:element>
</xs:sequence>
<xs:attribute name="ComponentResourceURL" type="xs:string"
            use="required"/>
</xs:complexType>
</xs:element>
```

```
    <xs:element name="Signatures">
     <xs:complexType>
     <xs:sequence>
      <xs:element name="Signature" type="ObjectSignatureType"/>
     </xs:sequence>
     </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:attribute name="Version" type="xs:string" use="required"/>
   </xs:complexType>
   </xs:element>
  </xs:sequence>
  </xs:complexType>
 </xs:element>

 </xs:schema>
```

The **ObjectHeaderType** and **ObjectSignatureType** are specified in sections 2.2.2.2.12 and 2.2.2.2.13. The **PKIPolicyType** is specified in section 2.2.2.2.14.

The following table describes the elements and attributes:

XPath	Description
/fragment	Object data fragment element
/fragment/ManagedObject/@Version	The attribute MUST be "0,0,0,0"
/fragment/ManagedObject/Header	Object header element
/fragment/ManagedObject/Body	Object body element
/fragment/ManagedObject/Body/@ComponentResourceURL	The value MUST be "http://components.groove.net/Groove/Components/Root.osd?Package=net.groove.Groove.SystemComponents.GrooveAccountMgr_DLL&Version=0&Factory=DevicePolicy"
/fragment/ManagedObject/Body/Policy	Policy element
/fragment/ManagedObject/Body/Policy/@Flags	The value MUST be one the following values or a value produced by using a bitwise OR operation on one or more of the following values: 0x04: Client cannot change communicator integration settings. 0x08: Disable public directory contact search.
/fragment/ManagedObject/Body/Policy/Bandwidth	Bandwidth element. Omit if no bandwidth policies.
/fragment/ManagedObject/Body/Policy/Bandwidth/@DisplayMask	Can define how to format the value/unit pair for display purpose. Examples: "%d Percent", "%d Megabits per second"
/fragment/ManagedObject/Body/Policy/Bandwidth/@Value	MAY contain bandwidth value.
/fragment/ManagedObject/Body	The value can be one of the following values:

XPath	Description
/Policy/Bandwidth/@Unit	0: bits per second 1: kilobits per second 2: megabits per second 3: percent
/fragment/ManagedObject/Body/Policy/Audit	Audit element. Omit if no audit policies.
/fragment/ManagedObject/Body/Policy/Audit/Client	Audit client element. Omit if no client policies.
/fragment/ManagedObject/Body/Policy/Audit/Client/@Flags	The value can be one of the following values or a value produced by using a bitwise OR operation on two or more of the following values: 0x00000001: Enable non-telespace auditing. The term **telespace** is a synonym for **shared space**. 0x00000002: Enable telespace auditing.
/fragment/ManagedObject/Body/Policy/WebServices	MUST be ignored by the client. The server omits if no WebService policies.
/fragment/ManagedObject/Body/Policy/WebServices/@IsDirectEnabled	MUST be ignored by the client.
/fragment/ManagedObject/Body/Policy/PKI	public key infrastructure (PKI) policy element. Omit if no PKI policy.
/fragment/ManagedObject/Body/Policy/AuditServer	Audit server element. Omit if no audit server policies.
/fragment/ManagedObject/Body/Policy/AuditServer/@Flags	The value can be one of the following values or a value produced by using a bitwise OR operation on two or more of the following values: 0x01: Audit file content 0x02: Audit instant messages and invitations 0x04: Audit login and logoff 0x08: Audit contact addition and deletion 0x10: Audit dynamics 0x20: Client will not run until auditing can be enabled 0x40: Audit telespace
/fragment/ManagedObject/Body/Policy/AuditServer/@ServerURL	Can contain Audit Server URL
/fragment/ManagedObject/Body/Policy/AuditServer/@UploadFrequency	Can contain upload interval in days.
/fragment/ManagedObject/Body/Policy/AuditServer/@UploadFrequency2	MAY contain upload interval in milliseconds.
/fragment/ManagedObject/Body/Policy/AuditServer/AuditItems	Audit items element. Omit if no audit items.
/fragment/ManagedObject/Body	Audit item element. Omit if no audit item.

XPath	Description
/Policy/AuditServer/AuditItem	
/fragment/ManagedObject/Body /Policy/AuditServer/AuditItem/ComponentURL	Audit component URL element
/fragment/ManagedObject/Signatures	Contains signature data element
/fragment/ManagedObject/Signatures/Signature	Signature data element

2.2.2.2.10.5 DomainTrustPolicy Object

The following specifies the **DomainTrustPolicy** object schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="fragment">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Header" type="ObjectHeaderType"/>
   <xs:element name="Body">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="Policy">
     <xs:complexType>
      <xs:sequence>
      <xs:element name="Item">
       <xs:complexType>
       <xs:attribute name="Certificate" type="xs:base64Binary" use="required"/>
       <xs:attribute name="InOrganization" type="xs:int" use="required"/>
       <xs:attribute name="Name" type="xs:string" use="required"/>
       </xs:complexType>
      </xs:element>
      </xs:sequence>
     </xs:complexType>
     </xs:element>
    </xs:sequence>
    <xs:attribute name="ComponentResourceURL" type="xs:string" use="required"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="Signatures">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="Signature" type="ObjectSignatureType"/>
    </xs:sequence>
    </xs:complexType>
   </xs:element>
   </xs:sequence>
   <xs:attribute name="Version" type="xs:string" use="required"/>
  </xs:complexType>
```

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

  </xs:schema>
```

The **ObjectHeaderType** and the **ObjectSignatureType** are specified in sections 2.2.2.2.12 and 2.2.2.2.13.

The following table describes the elements and attributes:

XPath	Description
/fragment	Object data fragment element
/fragment/ManagedObject/@Version	The attribute value MUST be "0,0,0,0"
/fragment/ManagedObject/Header	Object header element
/fragment/ManagedObject/Body	Object body element
/fragment/ManagedObject/Body/@ComponentResourceURL	The value MUST be "http://components.groove.net/Groove/Components/Root.osd?Package=net.groove.Groove.SystemComponents.GrooveAccountMgr_DLL&Version=0&Factory=DomainTrustPolicy"
/fragment/ManagedObject/Body/Policy	Policy element
/fragment/ManagedObject/Body/Policy/Item	Policy item
/fragment/ManagedObject/Body/Policy/Item/@Certificate	MUST contain cross-certified domain certificate.
/fragment/ManagedObject/Body/Policy/Item/@InOrganization	MUST be ignored by the client
/fragment/ManagedObject/Body/Policy/Item/@Name	MUST be ignored by the client
/fragment/ManagedObject/Signatures	Signatures element
/fragment/ManagedObject/Signatures/Signature	Signature element

2.2.2.2.10.6 Identity Object

The following specifies the **Identity** object schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="fragment">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Header" type="ObjectHeaderType"/>
   <xs:element name="Body">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="IdentityTemplate">
     <xs:complexType>
      <xs:sequence>
      <xs:element name="ManagementDomainMigration" minOccurs="0">
       <xs:complexType>
       <xs:attribute name="ServerURL" type="xs:string" use="required"/>
       </xs:complexType>
      </xs:element>
      <xs:element name="Contact">
       <xs:complexType>
       <xs:sequence>
        <xs:element name="vCard">
        <xs:complexType>
         <xs:attribute name="Data" type="xs:base64Binary" use="required"/>
        </xs:complexType>
        </xs:element>
        <xs:element name="RelayDevices">
        <xs:complexType>
         <xs:sequence>
         <xs:element name="RelayDevice">
          <xs:ComplexType>
          <xs:attribute name="AuthorizationToken" type="xs:string"/>
          <xs:attribute name="Certificate" type="xs:base64Binary"/>
          <xs:attribute name="URL" type="xs:string"/>
          </xs:ComplexType>
         </xs:element>
         </xs:sequence>
        </xs:complexType>
        </xs:element>
        <xs:element name="PresenceDevices">
        <xs:complexType>
         <xs:sequence>
         <xs:element name="PresenceDevice">
          <xs:ComplexType>
          <xs:attribute name="URL" type="xs:string"/>
          <xs:attribute name="Certificate" type="xs:base64Binary"/>
          <xs:attribute name="AuthorizationToken" type="xs:string"/>
          </xs:ComplexType>
         </xs:elment>
         </xs:sequence>
        </xs:complexType>
        </xs:element>
        <xs:element name="Certificate">
        <xs:complexType>
         <xs:sequence>
```

```
        <xs:element name="Certificate">
         <xs:complexType>
         <xs:attribute name="ExpirationDate"
                     type="xs:double" use="required"/>
         <xs:attribute name="SignerAddress" type="xs:string" use="required"/>
         <xs:attribute name="SignerKeyHash" type="xs:base64Binary"
                     use="required"/>
         <xs:attribute name="Signature" type="xs:base64Binary" use="required"/>
         </xs:complexType>
        </xs:element>
        </xs:sequence>
       </xs:complexType>
       </xs:element>
       <xs:element name="CustomFields">
       <xs:complexType>
        <xs:attribute name="_95_95Affiliation" type="xs:string" use="required"/>
        <xs:attribute name="_95_95_95Affiliation_95Flags" type="xs:int"
                    use="required"/>
       </xs:complexType>
       </xs:element>
      </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="Origin">
      <xs:complexType>
      <xs:sequence>
       <xs:element name="ManagementDomain">
       <xs:complexType>
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="DisplayName" type="xs:string" use="required"/>
        <xs:attribute name="ServerURL" type="xs:string" use="required"/>
        <xs:attribute name="Certificate" type="xs:base64Binary"
                    use="required"/>
       </xs:complexType>
       </xs:element>
      </xs:sequence>
      <xs:attribute name="Name" type="xs:string" use="required"/>
      </xs:complexType>
     </xs:element>
     </xs:sequence>
     <xs:attribute name="Flags" type="xs:int" use="required"/>
    </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="ComponentResourceURL" type="xs:string"
               use="required"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="Signatures">
   <xs:complexType>
   <xs:sequence>
    <xs:element name="Signature" type="ObjectSignatureType"/>
   </xs:sequence>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
  <xs:attribute name="Version" type="xs:string" use="required"/>
 </xs:complexType>
 </xs:element>
```

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

The **ObjectHeaderType** and the **ObjectSignatureType** are specified in sections 2.2.2.2.12 and 2.2.2.2.13.

The following table describes the elements and attributes:

XPath	Description
/fragment	Fragment element
/fragment/ManagedObject/@Version	The attribute MUST be "0,0,0,0".
/fragment/ManagedObject/Header	Object header element
/fragment/ManagedObject/Body	Body element
/fragment/ManagedObject/Body/@ComponentResourceURL	The value MUST be "http://components.groove.net/Groove/Components/Root.osd?Package=net.groove.Groove.SystemComponents.GrooveAccountMgr_DLL&Version=0&Factory=IdentityTemplate".
/fragment/ManagedObject/Body/IdentityTemplate	Identity template element
/fragment/ManagedObject/Body/IdentityTemplate/@Flags	The value MUST be one of the following values: 1: Valid member 3: Disabled member
/fragment/ManagedObject/Body/IdentityTemplate/ManagementDomainMigration	Migration domain element. Omit if no management domain migration.
/fragment/ManagedObject/Body/IdentityTemplate/ManagementDomainMigration/@ServerURL	MUST contain URL of the server to be migrated to.
/fragment/ManagedObject/Body/Contact	Contact element
/fragment/ManagedObject/Body/Contact/VCard	vCard element
/fragment/ManagedObject/Body/Contact/VCard/@Data	vCard data.
/fragment/ManagedObject/Body/Contact/Certificate	Certificate element
/fragment/ManagedObject/Body/Contact/Certificate/@ExpirationDate	Expiration date in Coordinated Universal Time (UTC)
/fragment/ManagedObject/Body/Contact/Certificate/@Signature	Signature
/fragment/ManagedObject/Body/Co	Management server URL

XPath	Description
ntact/Certificate/@SignerAddress	
/fragment/ManagedObject/Body/Contact/Certificate/@SignerKeyHash	SHA1 hash of the DER-encoded management server signature public key.
/fragment/ManagedObject/Body/Contact/RelayDevices	Relay devices element
/fragment/ManagedObject/Body/Contact/RelayDevices/RelayDevice	Relay device element
/fragment/ManagedObject/Body/Contact/RelayDevices/RelayDevice/@AuthorizationToken	Pre-authentication token
/fragment/ManagedObject/Body/Contact/RelayDevices/RelayDevice/@Certificate	Relay server's Simple Symmetric Transport Protocol (SSTP) certificate.
/fragment/ManagedObject/Body/Contact/RelayDevices/RelayDevice/@URL	Relay device URL.
/fragment/ManagedObject/Body/Contact/PresenceDevices	Presence server devices element
/fragment/ManagedObject/Body/Contact/PresenceDevices/PresenceDevice	Presence server device element
/fragment/ManagedObject/Body/Contact/PresenceDevices/PresenceDevice/@URL	Presence server device URL
/fragment/ManagedObject/Body/Contact/PresenceDevices/PresenceDevice/@Certificate	Presence server device certificate.
/fragment/ManagedObject/Body/Contact/PresenceDevices/PresenceDevice/@AuthorizationToken	Pre-authentication token
/fragment/ManagedObject/Body/Contact/CustomFields	Custom fields element
/fragment/ManagedObject/Body/Contact/CustomFields/@_95_95Affiliation	Domain affiliation of a member
/fragment/ManagedObject/Body/Contact/CustomFields/@_95_95_95Affiliation_95Flags	The value MUST be 0x4000000.
/fragment/ManagedObject/Body/Origin	Origin element
/fragment/ManagedObject/Body/Origin/@Name	The value MUST be "urn:groove.net:ManagementDomain".

XPath	Description
/fragment/ManagedObject/Body/Origin/ManagementDomain	Management domain element
/fragment/ManagedObject/Body/Origin/ManagementDomain/@Certificate	Domain certificate
/fragment/ManagedObject/Body/Origin/ManagementDomain/@DisplayName	Domain display name
/fragment/ManagedObject/Body/Origin/ManagementDomain/@Name	Domain GUID
/fragment/ManagedObject/Body/Origin/ManagementDomain/@ServerURL	Domain server URL
/fragment/ManagedObject/Signatures	Signatures element
/fragment/ManagedObject/Signatures/Signature	Signature element

2.2.2.2.10.7 IdentityPolicy Object

The following specifies the **IdentityPolicy** object schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="fragment">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Header" type="ObjectHeaderType"/>
   <xs:element name="Body">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="Policy">
     <xs:complexType>
      <xs:sequence>
      <xs:element name="Contact">
       <xs:complexType>
       <xs:sequence>
        <xs:element name="VCard" minOccurs="0">
        <xs:complexType>
         <xs:attribute name="ChangeFlags" type="xs:int" default="2"/>
        </xs:complexType>
        </xs:element>
        <xs:element name="Policies" minOccurs="0">
        <xs:complexType>
         <xs:sequence>
         <xs:element name="DirectoryListings" minOccurs="0">
```

```
        <xs:complexType>
        <xs:sequence>
         <xs:element name="DirectoryListing" minOccurs="0" maxOccurs="unbounded">
         <xs:complexType>
          <xs:attribute name="Name" type="xs:string" use="required"/>
          <xs:attribute name="Value" type="xs:string" use="required"/>
         </xs:complexType>
         </xs:element>
        </xs:sequence>
        </xs:complexType>
       </xs:element>
       </xs:sequence>
      </xs:complexType>
      </xs:element>
     </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="Backup" minOccurs="0">
     <xs:complexType>
     <xs:attribute name="Interval" type="xs:double" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Telespaces" minOccurs="0">
     <xs:complexType>
     <xs:attribute name="DefaultTemplateComponentResourceURL" type="xs:string"/>
     <xs:attribute name="MinimumTemplateComponentResourceURL" type="xs:string"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="PKI" type="PKIPolicyType" minOccurs="0"/>
    <xs:element name="DeviceManagement" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
     <xs:sequence>
      <xs:element name="ManagementDomain" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:attribute name="Certificate" type="xs:base64Binary"/>
       <xs:attribute name="DisplayName" type="xs:string"/>
       <xs:attribute name="Name" type="xs:string"/>
       <xs:attribute name="ReportingInterval" type="xs:int"/>
       <xs:attribute name="ReportingPolicy" type="xs:string"/>
       <xs:attribute name="ServerURL" type="xs:string"/>
      </xs:complexType>
      </xs:element>
     </xs:sequence>
     </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:attribute name="BlockedFileTypes" type="xs:string"/>
    <xs:attribute name="Flags" type="xs:int"/>
    <xs:attribute name="PeerAuthenticationLevel" type="xs:int"/>
     <xs:attribute name="RestrictedForestNames" type="xs:string" />
   </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="ComponentResourceURL" type="xs:string" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="Signatures">
  <xs:complexType>
  <xs:sequence>
```

```
      <xs:element name="Signature" type="ObjectSignatureType"/>
     </xs:sequence>
     </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:attribute name="Version" type="xs:string" use="required"/>
   </xs:complexType>
   </xs:element>
  </xs:sequence>
  </xs:complexType>
 </xs:element>

 </xs:schema>
```

The **ObjectHeaderType** and the **ObjectSignatureType** are specified in sections 2.2.2.2.12 and 2.2.2.2.13. The **PKIPolicyType** is specified in section 2.2.2.2.14.

The following table describes the elements and attributes:

XPath	Description
/fragment	Fragment element
/fragment/ManagedObject/@Version	The attribute MUST be "0,0,0,0".
/fragment/ManagedObject/Header	Object header element
/fragment/ManagedObject/Body	Object body element
/fragment/ManagedObject/Body/@ComponentResourceURL	The value MUST be "http://components.groove.net/Groove/Components/Root.osd?Package=net.groove.Groove.SystemComponents.GrooveAccountMgr_DLL&Version=0&Factory=IdenityPolicy".
/fragment/ManagedObject/Body/Policy	Policy element
/fragment/ManagedObject/Body/Policy/@BlockedFileTypes	MAY contain a comma delimited list of blocked file types.
/fragment/ManagedObject/Body/Policy/@RestrictedForestNames	MAY contain a space delimited list of operating system login forest names.
/fragment/ManagedObject/Body/Policy/@Flags	The value MUST be one of following values or a value produced by using bitwise OR operator on two or more of the following values : 0x0: Do not apply any of the following options. 0x1: Identity can only be used on devices managed by identity's management domain. 0x2: Automatically manage devices at account configuration.
/fragment/ManagedObject/Body/Policy/@PeerAuthenticationLevel	The value MUST be one of the following values: 0: Do not warn or restrict members when communicating with any contact. 1: Warn users when communicating with unauthenticated contacts. 2: Restrict users from communicating with unauthenticated contacts.
/fragment/ManagedObject/Body/Poli	Contact element

XPath	Description
cy/Contact	
/fragment/ManagedObject/Body/Policy/Contact/VCard	vCard element. Omit if no vCard policy.
/fragment/ManagedObject/Body/Policy/Contact/VCard/@ChangeFlags	The value MUST be: 0x02: Member cannot change vCard content.
/fragment/ManagedObject/Body/Policy/Contact/Policies	Policies element. Omit if no policies.
/fragment/ManagedObject/Body/Policy/Contact/Policies/DirectoryListings	Directory listings element. Omit if no directory listing polices.
/fragment/ManagedObject/Body/Policy/Contact/Policies/DirectoryListings/DirectoryListing	Directory listing element. Omit if no directory listing police.
/fragment/ManagedObject/Body/Policy/Contact/Policies/DirectoryListings/DirectoryListing/@Name	The value MUST be one of the following contact directory values: "$GrooveNet" : public domain contact directory. "$ManagementDomain" : management domain's contact directory.
/fragment/ManagedObject/Body/Policy/Contact/Policies/DirectoryListings/DirectoryListing/@Value	MUST be one of the following value: 0: User can decide to publish or not to publish vCard. 1: Automatically publish vCard. 2: Never publish vCard.
/fragment/ManagedObject/Body/Policy/Backup	Backup policy element. Omit if no backup policy.
/fragment/ManagedObject/Body/Policy/Backup/@Interval	Backup Interval value in milliseconds.
/fragment/ManagedObject/Body/Policy/Telespaces	Telespaces element. Omit if no telespace policy.
/fragment/ManagedObject/Body/Policy/Telespaces/@DefaultTemplateComponentResouceURL	Default telespace template component resource URL
ManagedObject/Body/Policy/Telespaces/@MinimumTemplateComponentResouceURL	Minimum version of telespace template component resource URL.
ManagedObject/Body/Policy/PKI	public key infrastructure (PKI) element. Omit if no PKI policy.
ManagedObject/Body/Policy/DeviceManagement	Device management element. Omit if no device management policies.
ManagedObject/Body/Policy/DeviceManagement/ManagementDomain	Device management domain element. MUST contain same domain information as the identity's domain. Omit if no management domain.
ManagedObject/Body/Policy/DeviceManagement/ManagementDomain/@Certificate	Domain certificate
ManagedObject/Body/Policy/DeviceManagement/ManagementDomain	The value MUST be "60".

XPath	Description
/@ReportingInterval	
ManagedObject/Body/Policy/DeviceManagement/ManagementDomain/@ServerURL	Domain server URL
/fragment/ManagedObject/Header/ManagementDomain/@ReportingPolicy	The value MUST be "Management".
/fragment/ManagedObject/Signatures	Signatures element
/fragment/ManagedObject/Signatures/Signature	Signature element

2.2.2.2.10.8 PassphrasePolicy Object

The following specifies the **PassphrasePolicy** object schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="fragment">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Header" type="ObjectHeaderType"/>
   <xs:element name="Body">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="Policy">
     <xs:complexType>
      <xs:sequence>
      <xs:element name="Age" minOccurs="0">
       <xs:complexType>
       <xs:attribute name="Max" type="xs:double"/>
       </xs:complexType>
      </xs:element>
      <xs:element name="History" minOccurs="0">
       <xs:complexType>
       <xs:attribute name="Count" type="xs:int"/>
       </xs:complexType>
      </xs:element>
      <xs:element name="Strength" minOccurs="0">
       <xs:complexType>
       <xs:attribute name="Flags" type="xs:int"/>
       <xs:attribute name="MinTotalChars" type="xs:int"/>
       </xs:complexType>
      </xs:element>
      <xs:element name="Lockout" minOccurs="0">
       <xs:complexType>
       <xs:attribute name="Flags" type="xs:int"/>
       </xs:complexType>
      </xs:element>
```

```
      <xs:element name="Reset" minOccurs="0">
       <xs:complexType>
       <xs:attribute name="Text" type="xs:string"/>
       </xs:complexType>
      </xs:element>
      <xs:element name="DelayLockout" minOccurs="0">
       <xs:complexType>
       <xs:attribute name="Duration" type="xs:int"/>
       <xs:attribute name="LockoutFlag" type="xs:int"/>
       <xs:attribute name="Threshold" type="xs:int"/>
       <xs:attribute name="Vector" type="xs:string"/>
       </xs:complexType>
      </xs:element>
      </xs:sequence>
      <xs:attribute name="Flags" type="xs:int"/>
     </xs:complexType>
     </xs:element>
    </xs:sequence>
    <xs:attribute name="ComponentResourceURL" type="xs:string" use="required"/>
    </xs:complexType>
   </xs:element>
   <xs:element name="Signatures">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="Signature" type="ObjectSignatureType"/>
    </xs:sequence>
    </xs:complexType>
   </xs:element>
   </xs:sequence>
   <xs:attribute name="Version" type="xs:string" use="required"/>
  </xs:complexType>
  </xs:element>
 </xs:sequence>
 </xs:complexType>
</xs:element>

</xs:schema>
```

The **ObjectHeaderType** and the **ObjectSignatureType** are specified in sections 2.2.2.2.12 and 2.2.2.2.13.

The following table describes the elements and attributes:

XPath	Description
/fragment	Fragment element
/fragment/ManagedObject/@Version	The attribute value MUST be "0,0,0,0".
/fragment/ManagedObject/Header	Object header element
/fragment/ManagedObject/Body	Object body element
/fragment/ManagedObject/Body/@ComponentReso	The value MUST be "http://components.groove.net/Groove/Components/Root.osd?Package=net.groo

XPath	Description
urceURL	ve.Groove.SystemComponents.GrooveAccountMgr_DLL&Version=0&Factory=PassphrasePolicy".
/fragment/ManagedObject/Body/Policy	Policy element
/fragment/ManagedObject/Body/Policy/@Flags	The value can be one of the following values: 0x00: Do not apply any of the following options. 0x01: Client cannot memorize Passphrase. 0x02: Client cannot use password hints.
/fragment/ManagedObject/Body/Policy/Age	Password age element. The server omits if no password age policy
/fragment/ManagedObject/Body/Policy/Age/@Max	Can contain maximum age of password in milliseconds.
ManagedObject/Body/Policy/History	Password history element. The server omits if no password history policy.
ManagedObject/Body/Policy/History/@Count	Can contain number of password to be maintained in password history list
ManagedObject/Body/Policy/Strength	Password strength element. The server omits if no password strength policy.
ManagedObject/Body/Policy/Strength/@Flags	The value can be one of following values or a value produced by using a bitwise OR operation on two or more of the following values: 0x00: Password does not need to satisfy any of the following requirements. 0x01: Password MUST contain at least one alpha character. 0x02: Password MUST contain at least one numeric character. 0x04: Password MUST contain mixed case alphanumeric characters. 0x08: Password MUST contain at least one punctuation symbol.
ManagedObject/Body/Policy/Strength/@MinTotalChars	Can contain minimum length of the Password
ManagedObject/Body/Policy/Lockout	MUST be ignored by the client. Omit if no password lockout policy.
ManagedObject/Body/Policy/Lockout/@Flags	MUST be ignored by the client.
ManagedObject/Body/Policy/Reset	Reset element. The server omits if no password reset policy.
ManagedObject/Body/Policy/Reset/@Text	Can contain instructions for manual password reset
ManagedObject/Body/Policy/DelayLockOut	Delay lockout element. The server omits if no delayed lockout policy.
ManagedObject/Body/Policy/DelayLockOut/@Duration	Can contain lockout duration in seconds

XPath	Description
ManagedObject/Body/Policy/DelayLockOut/@LockoutFlag	The value can be: 0x01: enable default lockout
ManagedObject/Body/Policy/DelayLockout/@Threshold	The value can be a value between 0 and 1000.
ManagedObject/Body/Policy/DelayLockOut/@Vector	See section 3.2.5.1.4 for details on generating delay lockout vector string.
/fragment/ManagedObject/Signatures	Signatures element
/fragment/ManagedObject/Signatures/Signature	Signature element

2.2.2.2.11 NumericType

The **NumericType** is extended from the xsd:int type:

```
<xs:complexType name="NumericType">
 <xs:simpleContent>
 <xs:extension base="xsd:int">
  <xs:attribute ref="xsi:type" use="required" fixed="xsd:int"/>
 </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

The following table describes the attribute of the type:

XPath	Description
/NumericType/@type	Specifies the data type. Its value MUST be "xsd:int".

2.2.2.2.12 ObjectHeaderType

The **ObjectHeaderType** specifies the data type for the managed object header:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:complexType name="ManagedObjectHeaderType">
  <xs:sequence>
  <xs:element name="ManagementDomain" type="ManagementDomainType"/>
 </xs:sequence>
 <xs:attribute name="IssuedTime" type="xs:double" use="required"/>
 <xs:attribute name="ReplacementPolicy" type="xs:string" use="required"/>
 <xs:attribute name="IntendedIdentityURL" type="xs:string" use="required"/>
 <xs:attribute name="GUID" type="xs:string" use="required"/>
 <xs:attribute name="Description" type="xs:string" use="required"/>
 <xs:attribute name="DisplayName" type="xs:string" use="required"/>
 <xs:attribute name="Name" type="xs:string" use="required"/>
```

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

The **ManagementDomainType** is specified in section 2.2.2.2.9.

The following table describes the elements and attributes:

XPath	Description
/ManagedObjectHeaderType/@Name	Policy name
/ManagedObjectHeaderType/@DisplayName	Policy display name
/ManagedObjectHeaderType/@Description	Policy description
/ManagedObjectHeaderType/@GUID	Object GUID
/ManagedObjectHeaderType/@IntendedIdentityURL	This attribute MUST be an empty string.
/ManagedObjectHeaderType/@IssuedTime	Object creation time, which MUST be in milliseconds since midnight 01/01/1970, represented as double
/ManagedObjectHeaderType/@ReplacementPolicy	This attribute MUST have the value: "$IssuedTime".

2.2.2.2.13 ObjectSignatureType

The **ObjectSignatureType** specifies the data type for the managed object signature:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:complexType name="ObjectSignatureType">
  <xs:attribute name="Fingerprint" type="xs:string" use="required"/>
  <xs:attribute name="Value" type="xs:base64Binary" use="required"/>
 </xs:complexType>
</xs:schema>
```

The following table describes the elements and attributes:

XPath	Description
/ObjectSignatureType/Signature	Signature element
/ObjectSignatureType/Signature/@Fingerprint	Fingerprint
/ObjectSignatureType/Signature/@Value	Signature value

2.2.2.2.14 PKIPolicyType

The **PKIPolicyType** specifies the data type for the PKI policy:

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

```
<xs:element name="Feature" minOccurs="0" maxOccurs="unbounded">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="CertificateType" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="Certificates" minOccurs="0">
    <xs:complexType>
    <xs:sequence>
     <xs:element name="Item" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="Value" type="xs:string"/>
     </xs:complexType>
     </xs:element>
    </xs:sequence>
    </xs:complexType>
   </xs:element>
   </xs:sequence>
   <xs:attribute name="ID" type="xs:int"/>
  </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="ID" type="xs:int"/>
 <xs:attribute name="Flags" type="xs:int"/>
 <xs:attribute name="KeyUsageFlags" type="xs:int"/>
 <xs:attribute name="RevocationFreshness" type="xs:int"/>
 </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="Flags" type="xs:int"/>
</xs:complexType>
```

The following table describes the elements and attributes:

XPath	Description
/PKIPolicyType/@Flags	The value MUST be 0x01.
/PKIPolicyType/Feature	PKI policy feature element. Omit if no feature policy.
/PKIPolicyType/Feature/@ID	The value MUST be 0x02.
/PKIPolicyType/Feature/@Flags	Flag bit value 0x01 indicates a required feature policy. Flag bit value 0x02 indicates to check revocations of certificates.
/PKIPolicyType/Feature/@KeyUsageFlags	MUST be ignored by the client.
/PKIPolicyType/Feature/@RevocationFreshness	Lag time in seconds for invalidating revoked certificates
/PKIPolicyType/Feature/CertificateType	Certificate type element. Omit if no certificate type policy.
/PKIPolicyType/Feature/CertificateType/@ID	Certificate type identifier
/PKIPolicyType/Feature/Certificates	Certificates element. Omit if no policies for certificates.

XPath	Description
/PKIPolicyType/Feature/Certificates/Item	Certificate item element. Omit if no certificate item.
/PKIPolicyType/Feature/Certificates/Item/@Value	Certificate item value, DER encoded

2.2.2.2.15 ServiceFaultResponseType

The **ServiceFaultResponseType** type contains a fault code and a fault string.

```
<xs:complexType name="ServiceFaultResponseType">
 <xs:sequence>
  <xs:element name="faultCode" type="xs:int"/>
  <xs:element name="faultString" type="xs:string"/>
 </xs:sequence>
 </xs:complexType>
```

The following table describes the elements of the type:

XPath	Description
/ServiceFaultResponseType/faultCode	Fault code element
/ServiceFaultResponseType/faultString	Fault string element

The following table lists service fault codes:

Fault Code	Descriptions
105	Malformed SOAP request
200	Account not found
201	Account verification failed
203	Error Processing event
204	A required parameter was missing or invalid.
205	Unknown security error processing event
207	Exception during contact fetch
208	Domain join failed.
209	Domain not found.
210	Re-enrollment is required.
211	User not found.
212	Account backup failed.
214	Audit file upload query failed.
215	Audit file upload failed.

Fault Code	Descriptions
216	Audit log upload failed.
217	Audit log upload query failed.
218	Password reset failed.
221	Invalid authentication method
222	Remote user name is empty.
225	No backups
226	Fetching backup failed.
227	Not configured for auto account configuration
401	Activation code invalid.
402	Activation code already enrolled.
403	Signature verification failure during enrollment

2.2.2.2.16 ServiceRequestType1

The **ServiceRequestType1** contains service request data using the **ContentPayloadType**.

```
<xs:complexType name="ServiceRequestType1">
 <xs:sequence>
 <xs:element name="Payload" type="ContentPayloadType"/>
 <xs:element name="Version" type="NumericType"/>
 <xs:element name="LastBroadcastProcessed" type="NumericType"/>
 <xs:element name="MessageSequenceNumber" type="NumericType"/>
 </xs:sequence>
</xs:complexType>
```

The **ContentPayloadType** is specified in section 2.2.2.2.5 and the **NumericType** is specified in section 2.2.2.2.11.

The following table describes the elements of the type:

XPath	Description
/ServiceRequestType1/Payload	Service request payload element
/ServiceRequestType1/Version	Client version element
/ServiceRequestType1/LastBroadcastProcessed	Last server command processed element
/ServiceRequestType1/MessageSequenceNumber	Message sequence number element

2.2.2.2.17 ServiceRequestType2

The **ServiceRequestType2** contains service request data using the **ContentPayloadType**.

```
<xs:complexType name="ServiceRequestType2>
 <xs:sequence>
 <xs:element name="Payload" type="ContentPayloadType"/>
 <xs:element name="Version" type="NumericType"/>
 <xs:element name="LastBroadcastProcessed" type="NumericType"/>
</xs:sequence>
</xs:complexType>
```

The **ContentPayloadType** is specified in section 2.2.2.2.5 and the **NumericType** is specified in section 2.2.2.2.11.

The following table describes the elements of the type:

XPath	Description
/ServiceRequestType2/Payload	Service request payload element
/ServiceRequestType2/Version	Client version element
/ServiceRequestType2/LastBroadcastProcessed	Last server command processed element

2.2.2.2.18 ServiceRequestType3

The **ServiceRequestType3** contains service request data using the **AttributePayloadType**.

```
<xs:complexType name="ServiceRequestType3>
 <xs:sequence>
 <xs:element name="Payload" type="AttributePayloadType"/>
 <xs:element name="Version" type="NumericType"/>
 </xs:sequence>
</xs:complexType>
```

The **AttributePayloadType** is specified in section 2.2.2.2.2 and the **NumericType** is specified in section 2.2.2.2.11.

The following table describes the elements of the type:

XPath	Description
/ServiceRequestType3/Payload	Service request payload element
/ServiceRequestType3/Version	Service version element

2.2.2.2.19 ServiceResponseType1

The **ServiceResponseType1** contains only a service status code.

```
<xs:complexType name="ServiceResponseType1">
 <xs:sequence>
 <xs:element name="ReturnCode" type="NumericType"/>
 </xs:sequence>
</xs:complexType>
```

The **NumericType** is specified in section 2.2.2.2.11.

The following table describes the element of the type:

XPath	Description
/ServiceResponseType1/ReturnCode	Service status code element (0 represents success)

2.2.2.2.20 ServiceResponseType2

The **ServiceResponseType2** contains a service status code and a response payload.

```
<xs:complexType name="ServiceResponseType2">
 <xs:sequence>
 <xs:element name="ReturnCode" type="NumericType"/>
 <xs:element name="Payload" type="AttributePayloadType"/>
</xs:sequence>
</xs:complexType>
```

The **AttributePayloadType** is specified in section 2.2.2.2.2 and the **NumericType** is specified in section 2.2.2.2.11.

The following table describes the elements of the type:

XPath	Description
/ServiceResponseType2/ReturnCode	Service status code element (0=Success)
/ServiceResponseType2 /Payload	Service response payload element

2.2.2.2.21 ServiceResponseType3

The **ServiceResponseType3** contains a service status code and a response payload.

```
<xs:complexType name="ServiceResponseType3">
 <xs:sequence>
 <xs:element name="ReturnCode" type="NumericType"/>
 <xs:element name="ManagedObjects" minOccurs="0" type="AttributePayloadType"/>
 </xs:sequence>
</xs:complexType>
```

The **AttributePayloadType** is specified in section 2.2.2.2.2 and the **NumericType** is specified in section 2.2.2.2.11.

The following table describes the elements of the type:

XPath	Description
/ServiceResponseType3/ReturnCode	Service status code element (0 represents success)
/ServiceResponseType3 /ManagedObjects	Managed object payload element. Omit if no managed objects.

2.2.2.2.22 SEType

The **SEType** contains encrypted and base64 encoded (as specified in [RFC4648]) service specific data. This type is used in the "urn:groove.net" namespace.

```
<xs:complexType name="SEType"/>
 <xs:sequence>
 <xs:element name="Enc">
  <xs:complexType>
  <xs:attribute name="EC" type="xs:base64Binary" use="required"/>
  <xs:attribute name="IV" type="xs:base64Binary" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="Auth">
  <xs:complexType>
  <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
  </xs:complexType>
 </xs:element>
 </xs:sequence>
</xs:complexType>
```

The following table describes the elements and attributes of the type:

XPath	Description
/SEType/Enc	Encrypted element
/SEType /Enc/@EC	Encrypted content
/SEType /Enc/@IV	Initialization vector for the encryption and decryption
/SEType /Auth	Authenticator element
/SEType /Auth/@MAC	**Message Authentication Code (MAC)**

2.2.2.3 Common Attributes

The following table summarizes the set of common XSD attribute definitions specified by this protocol. XSD attributes that are only used in a single message are specified in section 2.2.3.

Attribute	Description
data	Service payload data
DeviceGUID	Device GUID
DomainGUID	Domain GUID
GUID	Account GUID
GrooveVersion	Client version
HighPriority	Priority of the event (1=true, 0=false)
IdentityURL	Identity URL of the account
IsDeviceAccount	A Boolean value that MUST be true for a device account.

Attribute	Description
UniqueID	MUST be ignored by the server.
UserDeviceGuid	Device GUID
UserDeviceName	Client host name
EventID	Event identifier
_EventID	Event identifier
created	Message creation timestamp (seconds since midnight 01/01/1970).
EC	Encrypted content
IV	Initialization vector for the encryption and decryption
MAC	Message Authentication Code

2.2.2.3.1 data

The **data** attribute contains a service payload data.

```
<xs:attribute name="data" type="xs:base64Binary"/>
```

2.2.2.3.2 DeviceGUID

The **DeviceGUID** attribute contains a device GUID.

```
<xs:attribute name="DeviceGUID" type="xs:string"/>
```

2.2.2.3.3 DomainGUID

The **DomainGUID** attribute contains a domain GUID.

```
<xs:attribute name="DomainGUID" type="xs:string"/>
```

2.2.2.3.4 GUID

The **GUID** attribute contains an account GUID.

```
<xs:attribute name="GUID" type="xs:string"/>
```

2.2.2.3.5 GrooveVersion

The **GrooveVersion** attribute contains a client version string.

```
<xs:attribute name="GrooveVersion" type="xs:string"/>
```

2.2.2.3.6 HighPriority

The **HighPriority** attribute indicates if an event is a high priority event. For a high priority event, the **HighPriority** attribute MUST be set to 1.

```
<xs:attribute name="HighPriority" type="BooleanType"/>
```

2.2.2.3.7 IdentityURL

The **IdentityURL** attribute contains a user identity URL.

```
<xs:attribute name="IdentityURL" type="xs:string"/>
```

2.2.2.3.8 IsDeviceAccount

The **IsDeviceAccount** attribute indicates whether or not an account is a user account or device account. For a device account, the **IsDeviceAccount** attribute MUST be set to 1.

```
<xs:attribute name="IsDeviceAccount" type="BooleanType"/>
```

2.2.2.3.9 UniqueID

The **UniqueID** attribute MUST be ignored by the server.

```
<xs:attribute name="UniqueID" type="xs:string"/>
```

2.2.2.3.10 UserDeviceGuid

The **UserDeviceGuid** attribute contains a user device GUID.

```
<xs:attribute name="UserDeviceGuid" type="xs:string"/>
```

2.2.2.3.11 UserDeviceName

The **UserDeviceName** attribute contains the client host name.

```
<xs:attribute name="UserDeviceName" type="xs:string"/>
```

2.2.2.3.12 _EventID

The **_EventID** attribute contains an event identifier.

```
<xs:attribute name="_EventID" type="xs:int"/>
```

2.2.2.3.13 created

The **created** attribute contains a message creation timestamp.

```
<xs:attribute name="created" type="xs:int"/>
```

2.2.2.3.14 EC

The **EC** attribute contains encrypted service specific data.

```
<xs:attribute name="EC" type="xs:base64Binary"/>
```

2.2.2.3.15 IV

The **IV** attribute contains an initialization vector for encryption and decryption.

```
<xs:attribute name="IV" type="xs:base64Binary"/>
```

2.2.2.3.16 MAC

The **MAC** attribute contains a Message Authentication Code.

```
<xs:attribute name="MAC" type="xs:base64Binary"/>
```

2.2.3 Message Definitions

The following table lists the protocol messages:

Message	Description
AccountHeartbeat	Request for reporting account heartbeat
AccountHeartbeatResponse	Response for account heartbeat request
AccountStore	Request to store an account
AccountStoreResponse	Response for account store request
AuditLogUpload	Request for audit log upload
AuditLogUploadResponse	Response for audit log upload request
AuditLogUploadQuery	Request for the last audit log sequence number
AuditLogUploadQueryResponse	Response for audit log query request
AutoAccountCodeConfiguration	Request for auto account code configuration
AutoAccountCodeConfigurationResponse	Response for auto account code configuration request
AutoActivation	Request for an automatic activation for a user
AutoActivationResponse	Response for automatic activation request
AutomaticPasswordReset	Request for automatic password reset
AutomaticPasswordResetResponse	Response for automatic password reset request

Message	Description
ContactFetch	Request to fetch one or more contacts
ContactFetchResponse	Response for contact fetch request
ContactSearch	Request to search contacts
ContactSearchResponse	Response for contact search request
CreateAccount	Request to create an account
CreateAccountResponse	Response for create account request
DomainEnrollment	Request for domain enrollment
DomainEnrollmentResponse	Response for domain enrollment request
DomainMigrationStatus	Request for domain migration status
DomainMigrationStatusResponse	Response for domain migration status request
Enrollment	Request for member enrollment
EnrollmentResponse	Response for enrollment request
Failures	Request for reporting device management failures
FailuresResponse	Response for failures request
Fault	Fault response for a service request
FileUpload	Request to upload audit file
FileUploadResponse	Response for file upload request
FileUploadQuery	Request to test if an audit file has been uploaded
FileUploadQueryResponse	Response for file upload query request
IdentityPublish	Request to publish a member's contact
IdentityPublishResponse	Response for identity publish request
KeyActivation	Request for product or member activation
KeyActivationResponse	Response for key activation request
ManagedObjectInstall	Request for reporting a managed object installation
ManagedObjectInstallResponse	Response for managed object install request
ManagedObjectStatus	Request to test managed object status
ManagedObjectStatusResponse	Response for managed object status request
PassphraseResetRequest	Request for manual password reset
PassphraseResetRequestResponse	Response for manual passphrase reset request
PassphraseResetStatus	Request for manual passphrase reset status

Message	Description
PassphraseResetStatusResponse	Response for manual passphrase reset status request
StatisticsPackage	Request for reporting statistics package
StatisticsPackageResponse	Response for statistics package request
GMSConfig	Request for authenticated and non-authenticated server URLs.
GMSConfigResponse	Response for GMSConfig request.

2.2.3.1 AccountHeartbeat Message

The **AccountHeartbeat** message is a service request message.

```
<xs:element name="AccountHeartbeat" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.1.1 AccountHeartbeat Payload

The **/AccountHeartbeat/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The /fragment/Event/SE/Enc/@EC attribute data before encryption is specified as:

```
<xs:element name="AccountHeartbeat">
 <xs:complexType>
 <xs:attribute name="Version" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the elements and attributes:

XPath	Description
/AccountHeartbeat	Account heartbeat element
/AccountHeartbeat/@Version	Client version

2.2.3.2 AccountHeartbeatResponse Message

The **AccountHeartbeatResponse** message is the non-fault response message for the **AccountHeartbeat** service request message.

```
<xs:element name="AccountHeartbeatResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.3 AccountStore Message

The AccountStore message is a service request message.

```
<xs:element name="AccountStore" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.3.1 AccountStore Payload

The **/AccountStore/Payload** element contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Event"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The referenced **Event** element is specified as:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="Event">
 <xs:complexType>
  <xs:sequence>
  <xs:element xmlns:g="urn:groove.net" ref="g:SE"/>
  </xs:sequence>
  <xs:attribute name="BackupFragmenetSize" type="xs:int" use="required"/>
  <xs:attribute name="BackupFragmentCount" type="xs:int" use="required"/>
  <xs:attribute name="BackupGUID" type="xs:string"/>
  <xs:attribute name="BackupIndexCount" type="xs:int"  use="required"/>
  <xs:attribute name="BackupSize" type="xs:int" use="required"/>
  <xs:attribute name="BackupVersion" type="xs:int" use="required"/>
  <xs:attribute name="DeviceGUID" type="xs:string" use="required"/>
  <xs:attribute name="DomainGUID" type="xs:string" use="required"/>
  <xs:attribute name="GUID" type="xs:string" use="required"/>
  <xs:attribute name="GrooveVersion" type="xs:string" use="required"/>
  <xs:attribute name="IdentityURL" type="xs:string" use="required"/>
  <xs:attribute name="IsDeviceAccount" type="BooleanType" use="required"/>
  <xs:attribute name="PayloadAsStream" type="BooleanType" use="required"/>
  <xs:attribute name="UniqueID" type="xs:string" use="required"/>
  <xs:attribute name="UserDeviceGuid" type="xs:string" use="required"/>
  <xs:attribute name="UserDeviceName" type="xs:string" use="required"/>
  <xs:attribute name="_EventID" type="xs:int" use="required"/>
  <xs:attribute name="created" type="xs:int" use="required"/>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The "g:SE" element is specified in section 2.2.2.1.13. The **SE** element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

The following table describes the payload XML elements and attributes:

XPath	Description
/fragment/Event	Service event element
/fragment/Event/@BackupFragmentSize	Backup fragment size
/fragment/Event/@BackupFragmentCount	Backup fragment count
/fragment/Event/@BackupGUID	Backup GUID
/fragment/Event/@BackupIndexCount	Backup index count
/fragment/Event/@BackupSize	Backup size
/fragment/Event/@BackupVersion	Backup version
/fragment/Event/@DeviceGuid	Device GUID
/fragment/Event/@DomainGuid	Domain GUID
/fragment/Event/@GUID	Account GUID
/fragment/Event/@GrooveVersion	Client version
/fragment/Event/@IdentityURL	Identity URL of the account
/fragment/Event/@IsDeviceAccount	A Boolean value MUST be true for a device account
fragment/Event/@PayloadAsStream	PayloadAsStream MUST be set to true (1) and the payload MUST be a serialized binary stream object.
/fragment/Event/@UniqueID	MUST be ignored by the server
/fragment/Event/@UserDeviceGuid	Device GUID
/fragment/Event/@UserDeviceName	Client host name
/fragment/Event/@_EventID	Event identifier
/fragment/Event/@created	Message creation timestamp in Coordinated Universal Time (UTC)

The /fragment/Event/SE/Enc/@EC attribute contains data that has been encrypted and then Base64 encoded.

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Event"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
```

```
</xs:schema>
```

2.2.3.4 AccountStoreResponse Message

The **AccountStoreResponse** message is the non-fault response message for the AccountStore service request message.

```
<xs:element name="AccountStoreResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.5 AuditLogUpload Message

The **AuditLogUpload** message is a service request message.

```
<xs:element name="AuditLogUpload" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.5.1 AuditLogUpload Payload

The **/AuditLogUpload/Payload** element contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Event"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:element name="SE">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Auth">
   <xs:complexType>
   <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The referenced **Event** element is specified in the following schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="Event">
```

```
  <xs:complexType>
   <xs:sequence>
   <xs:element xmlns:g="urn:groove.net" ref="g:SE"/>"
   </xs:sequence>
   <xs:attribute name="BackupFragmentSize" type="xs:int"
  use="required"/>
   <xs:attribute name="BackupFragmentCount" type="xs:int" use="required"/>
   <xs:attribute name="BackupGUID" type="xs:string"/>
   <xs:attribute name="BackupIndexCount" type="xs:int" use="required"/>
   <xs:attribute name="BackupSize" type="xs:int" use="required"/>
   <xs:attribute name="BackupVersion" type="xs:int" use="required"/>
   <xs:attribute name="DomainGUID" type="xs:string" use="required"/>
   <xs:attribute name="GUID" type="xs:string" use="required"/>
   <xs:attribute name="GrooveVersion" type="xs:string" use="required"/>
   <xs:attribute name="IdentityURL" type="xs:string" use="required"/>
   <xs:attribute name="IsDeviceAccount" type="BooleanType" use="required"/>
   <xs:attribute name="UserDeviceGuid" type="xs:string" use="required"/>
   <xs:attribute name="UserDeviceName" type="xs:string" use="required"/>
   <xs:attribute name="_EA2" type="xs:base64Binary" use="required"/>
   <xs:attribute name="_EventID" type="xs:int" use="required"/>
   <xs:attribute name="created" type="xs:int" use="required"/>
  </xs:complexType>
  </xs:element>
 </xs:schema>
```

The referenced "g:SE" element is specified in the **fragment** element schema previously defined in this section. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

The following table describes the payload XML elements and attributes:

XPath	Description
/fragment/Event	Service event element
/fragment/Event/@BackupFragmentSize	Backup fragment size
/fragment/Event/@BackupFragmentCount	Backup fragment count
/fragment/Event/@BackupGUID	Backup GUID
/fragment/Event/@BackupIndexCount	Backup index count
/fragment/Event/@BackupSize	Backup size
/fragment/Event/@BackupVersion	Backup version
/fragment/Event/@DomainGuid	Domain GUID
/fragment/Event/@GUID	Account GUID
/fragment/Event/@GrooveVersion	Client version
/fragment/Event/@IdentityURL	Identity URL of the account
/fragment/Event/@IsDeviceAccount	A Boolean value MUST be true for a device account
/fragment/Event/@UserDeviceGuid	Device GUID

XPath	Description
/fragment/Event/@UserDeviceName	Client host name
/fragment/Event/@_EA2	Base64 encoded audit log data
/fragment/Event/@_EventID	Event identifier
/fragment/Event/@created	Message creation timestamp
/fragment/Event/SE	Secured element
/fragment/Event/SE/Auth	Authenticator element
/fragment/EventSE/Auth/@MAC	Message Authentication Code

The **/fragment/Event/@_EA2** attribute contains data specified as:

```
<xs:element name="LU">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="L">
  <xs:complexType>
   <xs:sequence>
   <xs:choice maxOccurs="unbounded">
    <xs:element name="BH">
    <xs:complexType>
     <xs:attribute name="_body" type="xs:base64Binary" use="required"/>
     <xs:attribute name="_iv" type="xs:base64Binary" use="required"/>
     <xs:attribute name="_key" type="xs:base64Binary" use="required"/>
     <xs:attribute name="_mac" type="xs:base64Binary" use="required"/>
    </xs:complexType>
    </xs:element>
    <xs:element maxOccurs="unbounded" name="E">
    <xs:complexType>
     <xs:attribute name="_body" type="xs:base64Binary" use="required"/>
     <xs:attribute name="_iv" type="xs:base64Binary" use="required"/>
     <xs:attribute name="_mac" type="xs:base64Binary" use="required"/>
    </xs:complexType>
    </xs:element>
    <xs:element name="EH">
    <xs:complexType>
     <xs:attribute name="_body" type="xs:base64Binary" use="required"/>
     <xs:attribute name="_iv" type="xs:base64Binary" use="required"/>
     <xs:attribute name="_mac" type="xs:base64Binary" use="required"/>
    </xs:complexType>
    </xs:element>
    <xs:element name="UM">
    <xs:complexType>
     <xs:attribute name="_dt" type="xs:string" use="required"/>
     <xs:attribute name="_q" type="xs:int" use="required"/>
    </xs:complexType>
    </xs:element>
   </xs:choice>
   </xs:sequence>
  </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="_dg" type="xs:string" use="required"/>
```

```
  <xs:attribute name="_dgh" type="xs:base64Binary" use="required"/>
  <xs:attribute name="_iv" type="xs:base64Binary" use="required"/>
  <xs:attribute name="_lok" type="xs:base64Binary" use="required"/>
  <xs:attribute name="_mac" type="xs:base64Binary" use="required"/>
  <xs:attribute name="_v" type="xs:string" use="required"/>
  </xs:complexType>
 </xs:element>
```

The following table describes the elements and attributes:

XPath	Description
/LU	Log upload element
/LU/@_dg	Device GUID
/LU/@_dgh	SHA1 hash of the Device GUID.
/LU/@_iv	Initialization vector for the encryption and decryption
/LU/@_lok	Lock on key
/LU/@_mac	Message Authentication Code
/LU/@_v	The value MUST be "1,0".
/LU/L	Log data element
/LU/L/BH	Begin header element
/LU/L/BH/@_body	Base64 encoded log data
/LU/L/BH/@_iv	Initialization vector for the encryption and decryption
/LU/L//BH/@_key	Encryption key
/LU/L/BH/@_mac	Message Authentication Code
/LU/L/E	Entry element
/LU/L/E/@_body	Base64 encoded log data
/LU/L/E/@_iv	Initialization vector for the encryption and decryption
/LU/L/E/@_mac	Message Authentication Code
/LU/L/EH	End header element
/LU/L/EH/@_body	Base64 encoded log data
/LU/L/EH/@_iv	Initialization vector for the encryption and decryption
/LU/L/EH/@_mac	Message Authentication Code
/LU/L/UM	Upload marker element
/LU/L/UM/@_dt	Date time in MM/DD/YYYY hh:mm:ss format where: MM=Month, DD=Day, YYYY=year, hh=hour, mm=minute, ss=second
/LU/L/UM/@_q	Upload sequence number (current _q = last uploaded _q + 1)

2.2.3.6 AuditLogUploadResponse Message

The **AuditLogUploadResponse** message is the non-fault response message for the **AuditLogUpload** service request message.

```
<xs:element name="AuditLogUploadResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.6.1 AuditLogUploadResponse Payload

The **/AuditLogUploadResponse/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute contains payload data specified as:

```
<xs:element name="LUR">
 <xs:complexType>
 <xs:attribute name="_com" type="BooleanType" use="required"/>
 <xs:attribute name="_dgh" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_lrh" type="xs:int" use="required"/>
 <xs:attribute name="_lrl" type="xs:int" use="required"/>
 <xs:attribute name="_sig" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_v" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the attributes:

XPath	Description
/LUR	Log upload response element
/LUR/@_com	Indicates if an upload has finished (1=true, 0=false)
/LUR/@_dgh	Device GUID hash
/LUR/@_lrh	Log range high value

XPath	Description
/LUR/@_lrl	Log range low value
/LUR/@_sig	Signature
/LUR/@_v	Version. The value MUST be "1,0".

2.2.3.7 AuditLogUploadQuery Message

The **AuditLogUploadQuery** message is a service request message.

```
<xs:element name="AuditLogUploadQuery" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.7.1 AuditLogUploadQuery Payload

The **/AuditLogUploadQuery/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="AuditLogUploadQuery">
 <xs:complexType>
 <xs:attribute name="_v" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the elements and attributes:

XPath	Description
/AuditLogUploadQuery	Audit log upload service element
/AuditLogUploadQuery/@_v	Version, its value MUST be "1,0".

2.2.3.8 AuditLogUploadQueryResponse Message

The **AuditLogUploadQueryResponse** message is the non-fault response message for the **AuditLogUploadQuery** service request message.

```
<xs:element name="AuditLogUploadQueryResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.8.1 AuditLogUploadQueryResponse Payload

The **/AuditLogUploadQueryResponse/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="LUQR">
 <xs:complexType>
 <xs:attribute name="_com" type="BooleanType" use="required"/>
 <xs:attribute name="_dgh" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_lrh" type="xs:int" use="required"/>
 <xs:attribute name="_lrl" type="xs:integer" use="required"/>
 <xs:attribute name="_sig" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_v" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the attributes:

XPath	Description
/LUQR	Log upload query response element
/LUR/@_com	Indicates if an upload has finished (1=true, 0=false)
/LUR/@_dgh	Device GUID hash
/LUR/@_lrh	Log high range
/LUR/@_lrl	Log low range
/LUR/@_sig	Signature
/LUR/@_v	Version, the value MUST be "1,0".

2.2.3.9 AutoAccountCodeConfiguration Message

The **AutoAccountCodeConfiguration** message is a service request message.

```
<xs:element name="AutoAccountCodeConfiguration" type="ServiceRequestType3"/>
```

The **ServiceRequestType3** is specified in section 2.2.2.2.18.

2.2.3.9.1 AutoAccountCodeConfiguration Payload

The **/AutoAccountCodeConfiguration/Payload/@data** attribute contains the payload data specified as:

```
<xs:element name="PayloadWrapper">
 <xs:complexType>
 <xs:attribute name="GrooveVersion" type="xs:string" use="required"/>
 <xs:attribute name="IsDomainMemberMigration" type="BooleanType"
use="required"/>
 <xs:attribute name="UserDeviceName" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the attributes:

XPath	Description
/PayloadWrapper	Element contains payload data element.
/PayloadWrapper/@GrooveVersion	Client version
/PayloadWrapper/@IsDomainMemberMigration	Indicates domain migration (1 represents true, 0 represents false).
/PayloadWrapper/@UserDeviceName	Client host name

2.2.3.10 AutoAccountCodeConfigurationResponse Message

The **AutoAccountCodeConfigurationResponse** message is the non-fault response message for the **AutoAccountCodeConfiguration** service request message.

```
<xs:element name="AutoAccountCodeConfigurationResponse"
            type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.10.1 AutoAccountCodeConfigurationResponse Payload

The **/AutoAccountCodeConfigurationResponse/Payload/@data** attribute contains the payload data. The following are the two types of payload data:

- For new account configuration, the payload data is specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="KeyActivation"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
```

```
</xs:schema>
```

The **KeyActivation** element is specified in section 2.2.2.1.4.

- For an account that has already been configured, the payload MUST use the following template:

```
<ReturnPayload
 BackupFragmentCount=[[- fragmentCount -]]
 BackupFragment_1=[[- fragment_1 -]]
 BackupFragment_2=[[- fragment_1 -]]
 …
 BackupFragment_n=[[- fragment_n -]]
/>
```

[[- fragmentCount -]]: The backup fragment count, specified as an integer.

[[- fragment_1 -]] … [[- fragment_n -]]]: The backup fragment data, Base64 encoded. The number of the backup fragment MUST be equal to the "BackupFragmentCount".

2.2.3.11 AutoActivation Message

The **AutoActivation** message is a service request message.

```
<xs:element name="AutoActivation" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.11.1 AutoActivation Payload

The **/AutoActivation/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="AutoActivationRequest">
 <xs:complexType>
 <xs:attribute name="AccountGUID" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

2.2.3.12 AutoActivationResponse Message

The **AutoActivationResponse** message is the non-fault response message for the **AutoActivation** service request message.

```
<xs:element name="AutoActivationResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.12.1 AutoActivationResponse Payload

The **/AutoActivationResponse/Payload/@data** attribute contains data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="ReturnPayload">
 <xs:complexType>
 <xs:attribute name="data" type="xs:base64Binary" use="required"/>
</xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/ReturnPayload	Returns payload data element
/ReturnPayload/@data	Contains return payload data

The value of the **/ReturnPayload/@data** attribute is the payload data in a XML fragment:

```
<xs:element name="KeyActivation">
 <xs:complexType>
 <xs:attribute name="ActivationKey" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/KeyActivation	Key activation element
/KeyActivation/@ActivationKey	Account configuration code

2.2.3.13 AutomaticPasswordReset Message

The **AutomaticPasswordReset** message is a service request message.

```
<xs:element name="AutomaticPasswordReset" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.13.1 AutomaticPasswordReset Payload

The **/AutomaticPasswordReset/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="AutomaticPasswordResetRequest"/>
   <xs:complexType>
   <xs:attribute name="AutoPasswordResetBody" type="xs:string"
       use="required"/>
   <xs:attribute name="AutoPasswordResetSalutation" type="xs:string"
       use="required"/>
   <xs:attribute name="AutoPasswordResetSubject" type="xs:string"
       use="required"/>
   <xs:attribute name="CertificatePublicKeyHash" type="xs:base64Binary"
       use="required"/>
   <xs:attribute name="ContactDisplayName" type="xs:string" use="required"/>
   <xs:attribute name="DigestAlgorithm" type="xs:string" use="required"/>
   <xs:attribute name="EmailAddress" type="xs:string" use="required"/>
   <xs:attribute name="EncryptedMasterKey" type="xs:base64Binary"
       use="required"/>
   <xs:attribute name="EncryptedSecretMasterKey" type="xs:base64Binary"
       use="required"/>
   <xs:attribute name="GUID" type="xs:string" use="required"/>
   <xs:attribute name="URL" type="xs:string" use="required"/>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The following table describes the XML elements and attributes:

XPath	Description
/fragment/AutomaticPasswordResetRequest	Automatic password reset request element
/fragment/ AutomaticPasswordResetRequest/@AutoPasswordResetBody	Password reset instruction that will be displayed to the user

XPath	Description
/fragment/ AutomaticPasswordResetRequest/@AutoPasswordResetSalutation	Automatic password reset e-mail salutation
/fragment/ AutomaticPasswordResetRequest/@AutoPasswordResetSubject	Automatic password reset e-mail subject
/fragment/ AutomaticPasswordResetRequest/@CertificatePublicKeyHash	SHA1 hash of the DER-encoded public key from the domain's data recovery certificate.
/fragment/ AutomaticPasswordResetRequest/@ContactDisplayName	Contact display name
/fragment/ AutomaticPasswordResetRequest/@DigestAlgorithm	Digest algorithm
/fragment/ AutomaticPasswordResetRequest/@EmailAddress	E-mail address
/fragment/ AutomaticPasswordResetRequest/@EncryptedMasterKey	Encrypted master key
/fragment/ AutomaticPasswordResetRequest/@EncryptedSecretMasterKey	Encrypted secret master key
/fragment/ AutomaticPasswordResetRequest/@GUID	Account GUID
/fragment/ AutomaticPasswordResetRequest/@URL	Identity URL

2.2.3.14 AutomaticPasswordResetResponse Message

The **AutomaticPasswordResetResponse** message is the non-fault response message for the **AutomaticPasswordReset** service request message.

```
<xs:element name="AutomaticPasswordResetResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.14.1 AutomaticPasswordResetResponse Payload

The **/AutomaticPasswordResetResponse/Payload/@data** attribute contains data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute contains data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="AutomaticPasswordResetRequest">
   <xs:complexType>
   <xs:attribute name="EmailAddress" type="xs:string" use="required"/>
   <xs:attribute name="EncryptedMasterKey" type="xs:base64Binary" use="required"/>
   <xs:attribute name="EncryptedMasterKeyIV" type="xs:base64Binary"
                 use="required"/>
   <xs:attribute name="EncryptedSecretMasterKey" type="xs:base64Binary"
                 use="required"/>
   <xs:attribute name="EncryptedSecretMasterKeyIV" type="xs:base64Binary"
                 use="required"/>
   <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
   <xs:attribute name="URL" type="xs:string" use="required"/>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The following table describes the XML elements and attributes:

XPath	Description
/fragment/AutomaticPasswordResetRequest	Automatic password reset request element
/fragment/AutomaticPasswordResetRequest/@EmailAddress	User's e-mail address
/fragment/AutomaticPasswordResetRequest/@EncryptedMasterKey	Encrypted master key
/fragment/AutomaticPasswordResetRequest/@EncryptedMasterKeyIV	Encrypted master key IV
/fragment/AutomaticPasswordResetRequest/@EncryptedSecretMasterKey	Encrypted secret master key
/fragment/AutomaticPasswordResetRequest/@EncrytedSecretMasterKeyIV	Encrypted secret master key IV
/fragment/AutomaticPasswordResetRequest/@MAC	Message Authentication Code
/fragment/AutomaticPasswordResetRequest/@URL	User identity URL

2.2.3.15 ContactFetch Message

The **ContactFetch** message is a service request message.

```
<xs:element name="ContactFetch" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.15.1 ContactFetch Payload

The **/ContactFetch/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="ContactFetch">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="IdentityList">
  <xs:complexType>
   <xs:sequence>
   <xs:element name="IdentityList" maxOccurs="unbounded">
    <xs:complexType>
    <xs:attribute name="IdentityGUID" type="xs:string" use="required"/>
    </xs:complexType>
   </xs:element>
   </xs:sequence>
  </xs:complexType>
  </xs:element>
 </xs:sequence>
 </xs:complexType>
</xs:element>
```

The following table describes the elements and attributes:

XPath	Description
/ContactFetch	Contact fetch element
/IdentityList	Identity list element
/IdentityList/IdentityList	Identity list element
/IdentityList/IdentityList/@IdentityGUID	Identity GUID

2.2.3.16 ContactFetchResponse Message

The **ContactFetchResponse** message is the non-fault response message for the **ContactFetch** service request message.

```
<xs:element name="ContactFetchResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.16.1 ContactFetchResponse Payload

The **/ContactFetchResponse/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
```

```
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="ReturnPayload">
 <xs:attribute name="Data" type="xs:base64Binary" use="required"/>
</xs:element>
```

Data can be decoded as UTF8 to an XML string. the schema of the decoded XML string is:

```
<xs:element name="IdentityList">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="Identity" maxOccurs="unbounded">
  <xs:complexType>
   <xs:attribute name="IdentityGUID" type="xs:string" use="required"/>
   <xs:attribute name="VCard" type="xs:base64Binary" use="required"/>
  </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="IdentityCount" type="xs:int" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/IdentityList	Identity list element
/IdentityList/@IdentityCount	Count of identities in the list
/IdentityList/Identity	Identity element
/IdentityList/Identity/@IdentityGUID	Identity GUID
/IdentityList/Identity/@VCard	Base64 encoded data

2.2.3.17 ContactSearch Message

The **ContactSearch** message is a service request message.

```
<xs:element name="ContactSearch" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.17.1 ContactSearch Payload

The **/ContactSearch/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="ContactSearch">
 <xs:complexType>
 <xs:attribute name="Query" type="xs:base64Binary" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the elements and attributes:

XPath	Description
/ContactSearch	Contact search element
/ContactSearch/@Query	Contact search query (any Unicode string value).

2.2.3.18 ContactSearchResponse Message

The **ContactSearchResponse** message is the non-fault response message for the **ContactSearch** service request message.

```
<xs:element name="ContactSearchResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.18.1 ContactSearchResponse Payload

The **/ContactSearchResponse/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="ReturnPayload">
  <xs:attribute name="Data" type="xs:base64Binary" use="required"/>
</xs:element>
```

Data can be decoded as UTF8 to an XML string. The schema of the decoded XML string is:

```
<xs:element name="ContactSearchResponse">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="Contact" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
   <xs:attribute name="City" type="xs:string" use="required"/>
   <xs:attribute name="CompanyEMail" type="xs:string" use="required"/>
   <xs:attribute name="EMail" type="xs:string" use="required"/>
   <xs:attribute name="FirstName" type="xs:string" use="required"/>
   <xs:attribute name="FullName" type="xs:string" use="required"/>
   <xs:attribute name="IdentityGUID" type="xs:string" use="required"/>
   <xs:attribute name="IdentityURL" type="xs:string" use="required"/>
   <xs:attribute name="LastName" type="xs:string" use="required"/>
   <xs:attribute name="State" type="xs:string" use="required"/>
  </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="Count" type="xs:int" use="required"/>
 <xs:attribute name="Max" type="xs:int" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/ContactSearchResponse	Contact search response element
/ContactSearchResponse/@Count	Contact count
/ContactSearchResponse/@Max	Maximum contact count
/ContactSearchResponse/Contact	Contact element. Omit if no contact found in a search result.
/ContactSearchResponse/Contact/@City	City of a contact
/ContactSearchResponse/Contact/@CompanyEmail	Company e-mail of a contact
/ContactSearchResponse/Contact/@FirstName	First name of a contact
/ContactSearchResponse/Contact/@FullName	Full name of a contact
/ContactSearchResponse/Contact/@IdentityGUID	Identity GUID of a contact
/ContactSearchResponse/Contact/@IdentityURL	Identity URL of a contact
/ContactSearchResponse/Contact/@LastName	Last name of a contact
/ContactSearchResponse/Contact/@State	State/Province of a contact

2.2.3.19 CreateAccount Message

The **CreateAccount** message is a service request message.

```
<xs:element name="CreateAccount" type="ServiceRequestType2"/>
```

The **ServiceRequestType2** is specified in section 2.2.2.2.17.

2.2.3.19.1 CreateAccount Payload

The **/CreateAccount/Payload** element contains the payload specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="Event"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:element name="SE">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Cert">
   <xs:complexType>
   <xs:attribute name="EPKAlgo" type="xs:string" use="required" fixed="DH"/>
   <xs:attribute name="EPubKey" type="xs:base64Binary" use="required"/>
   <xs:attribute name="EncAlgo" type="xs:string" use="required"/>
   <xs:attribute name="SPKAlgo" type="xs:string" use="required" fixed="RSA"/>
   <xs:attribute name="SPubKey" type="xs:base64Binary" use="required"/>
   <xs:attribute name="SigAlgo" type="xs:string" use="required" fixed="RSA"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="Auth">
   <xs:complexType>
   <xs:attribute name="Sig" type="xs:base64Binary" use="required"/>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
  <xs:attribute name="CSMKey" type="xs:base64Binary" use="required"/>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The referenced **Event** element is specified as:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="Event">
 <xs:complexType>
  <xs:sequence>
  <xs:element xmlns:g="urn:groove.net" ref="g:SE"/>
```

```
    </xs:sequence>
    <xs:attribute name="DomainGUID" type="xs:string" use="required"/>
    <xs:attribute name="Encrypted" type="BooleanType" use="required"/>
    <xs:attribute name="GUID" type="xs:string" use="required"/>
    <xs:attribute name="IsDeviceAccount" type="BooleanType" use="required"/>
    <xs:attribute name="created" type="xs:int" use="required"/>
   </xs:complexType>
   </xs:element>
  </xs:schema>
```

The "g:SE" element is specified in the **fragment** element schema previously defined in this section. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

The following table describes the payload XML elements and attributes:

XPath	Description
/fragment/Event	Service event element
/fragment/Event/@DomainGuid	Domain GUID
/fragment/Event/@Encrypted	Indicates whether the content is encrypted or not
/fragment/Event/@GUID	Account GUID
/fragment/Event/@IsDeviceAccount	A Boolean value MUST be true for a device account
/fragment/Event/@created	Message creation timestamp
/fragment/Event/SE	SE element
/fragment/Event/SE/@CSMKey	Secret key to be shared by the client and the server
/fragment/EventSE/Auth	Authenticator element
/fragment/EventSE/Auth/@Sig	Message signature
/fragment/Event/SE/Cert	Public key information element
/fragment/Event/SE/Cert/@EPKAlgo	Encryption public key algorithm. The value MUST be "DH".
/fragment/Event/SE/Cert/@EPubKey	Encryption public key, DER encoded
/fragment/Event/SE/Cert/@EncAlgo	Encryption algorithm. The value MUST be "RSA" or "ELGAMAL".
/fragment/Event/SE/Cert/@SPKAlog	Signature public key algorithm. The value MUST be "RSA".
/fragment/Event/SE/Cert/@SPubKey	Signature public key, DER encoded
/fragment/Event/SE/Cert/@SigAlgo	Signature algorithm. The value MUST be "RSA".

2.2.3.20 CreateAccountResponse Message

The **CreateAccountResponse** message is the non-fault response message for the **CreateAccount** service request message.

```
<xs:element name="CreateAccountResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.20.

2.2.3.21 DomainEnrollment Message

The **DomainEnrollment** message is a service request message.

```
<xs:element name="DomainEnrollment" type="ServiceRequestType3"/>
```

The **ServiceRequestType3** is specified in section 2.2.2.2.18.

2.2.3.21.1 DomainEnrollment Payload

The **/DomainEnrollment/Payload/@data** attribute contains the payload specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="PayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **PayloadWrapper** element is specified in section 2.2.2.1.9.

The **/fragment/PayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="Payload">
 <xs:complexType>
  <xs:attribute name="Contact" type="xs:base64Binary">
  <xs:attribute name="AccountGuid" type="xs:string">
  <xs:attribute name="ActivationKeySignature" type="xs:base64Binary">
  <xs:attribute name="GrooveVersion" type="xs:string">
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/Payload	Payload element
/Payload/@Contact	Contact data
/Payload/@AccountGuid	Account GUID
/Payload/@ActivationKeySignature	Activation key signature

XPath	Description
/Payload/@GrooveVersion	Client version

```
The /Payload/@Contact attribute data is specified as:
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Contact" type="ContactType"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ContactType** is specified in section 2.2.2.2.4.

2.2.3.22 DomainEnrollmentResponse Message

The **DomainEnrollmentResponse** message is the non-fault response message for the **DomainEnrollment** service request message.

```
<xs:element name="DomainEnrollmentResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.22.1 DomainEnrollmentResponse Payload

The **/DomainEnrollmentResponse/Payload/@data** attribute contains the payload specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
```

```
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="DomainEnrollment">
  <xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **DomainEnrollment** element is specified in the following schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="DomainEnrollment">
 <xs:complexType>
  <xs:sequence>
  <xs:element xmlns:g="urn:groove.net" ref="g:ManagementDomain"/>
  <xs:element name="ManagedObjects">
   <xs:complexType>
   <xs:sequence>
    <xs:element name="ManagedObject" type="ManagedObjectType"
        maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="Count" type="xs:int"/>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The "g:ManagementDomain" element is specified in section 2.2.2.1.6. The ManagementDomain element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net". The **ManagedObjectType** is specified in section 2.2.2.2.10.

The following table describes the XML elements and attributes:

XPath	Description
/fragment	Fragment element
/fragment/DomainEnrollment	Domain enrollment element
/fragment/DomainEnrollment/ManagementDomain	Management domain element
/fragment/DomainEnrollment/fragment/ManagedObjects	Managed objects element
/fragment/DomainEnrollment/fragment/ManagedObjects/@Count	The count of the objects
/fragment/DomainEnrollment/ManagedObjects/fragment/ManagedObject	Managed object element

2.2.3.23 DomainMigrationStatus Message

The **DomainMigrationStatus** message is a service request message.

```
<xs:element name="DomainMigrationStatus" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.23.1 DomainMigrationStatus Payload

The **/DomainMigrationStatus/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="MemberMigrationStatus">
 <xs:complexType>
  <xs:attribute name="MigratedDomainName" type="xs:string" use="required"/>
  <xs:attribute name="MigrationStatus" type="xs:int" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/MemberMigrationStatus	Member migration status element
/MemberMigrationStatus/@MigratedDomainName	Migrated domain name
/MemberMigrationStatus/@MigrationStatus	Migration status where 0 indicates success and all other values indicate failures.

2.2.3.24 DomainMigrationStatusResponse Message

The **DomainMigrationStatusResponse** message is the non-fault response message for the **DomainMigrationStatus** service request message.

```
<xs:element name="DomainMigrationStatusResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.25 Enrollment Message

The **Enrollment** message is a service request message.

```
<xs:element name="Enrollment" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.25.1 Enrollment Payload

The **/Enrollment/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="Payload">
 <xs:complexType>
 <xs:attribute name="Contact" type="xs:base64Binary" use="required"/>
 </xs:complexType>
</xs:element>
```

The **/Payload/@Contact** attribute contains data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Contact" type="ContactType"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ContactType** is specified in section 2.2.2.2.4.

The following table describes the elements and attributes:

XPath	Description
/fragment	Fragment element
/fragment/Contact	Contact element

2.2.3.26 EnrollmentResponse Message

The **EnrollmentResponse** message is the non-fault response message for the **Enrollment** service request message.

```
<xs:element name="EnrollmentResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.27 Failures Message

The **Failures** message is a service request message.

```
<xs:element name="Failures" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.27.1 Failures Payload

The **/Failures/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="Failures">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="Failure" maxOccurs="unbounded">
  <xs:complexType>
   <xs:attribute name="Subtype" type="xs:int" use="required"/>
   <xs:attribute name="Type" type="xs:int" use="required"/>
  </xs:complexType>
  </xs:element>
 </xs:sequence>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/Failures	Failures elements
/Failures/Failure	Failure element
/Failures/Failure/@SubType	The value MUST be one of the following values: 2: Unable to delete / modify registry entry. 3: Unable to set create registry entry. 5: Unable to retrieve device policies from the server. 6: User declined to manage the device.
/Failures/Failure/@Type	The value MUST be 6001.

2.2.3.28 FailuresResponse Message

The **FailuresResponse** message is the non-fault response message for the **Failures** service request message.

```
<xs:element name="FailuresResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.29 Fault Message

The **Fault** message is a Groove **SOAP fault** response message. It is used for any service request fault.

```
<xs:element name="Fault" type="ServiceFaultResponseType"/>
```

The **ServiceFaultResponseType** is specified in section 2.2.2.2.15.

2.2.3.30 FileUpload Message

The **FileUpload** message is a service request message.

```
<xs:element name="FileUpload" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.30.1 FileUpload Payload

The **/FileUpload/Payload** element contains the payload specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="Event"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:element name="SE">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Auth">
   <xs:complexType>
   <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The referenced **Event** element is specified in the following schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="Event">
 <xs:complexType>
  <xs:sequence>
  <xs:element xmlns:g="urn:groove.net" ref="g:SE"/>
  <xs:element name="AuditElemUploadXlink" type="FileUploadXlinkType"/>
  </xs:sequence>
  <xs:attribute name="BackupFragmenetSize" type="xs:int" use="required"/>
  <xs:attribute name="BackupFragmentCount" type="xs:int" use="required"/>
  <xs:attribute name="BackupGUID" type="xs:string" use="required"/>
  <xs:attribute name="BackupIndexCount" type="xs:int" use="required"/>
  <xs:attribute name="BackupSize" type="xs:int" use="required"/>
  <xs:attribute name="BackupVersion" type="xs:int" use="required"/>
  <xs:attribute name="DomainGUID" type="xs:string" use="required"/>
  <xs:attribute name="GUID" type="xs:string" use="required"/>
  <xs:attribute name="GrooveVersion" type="xs:string" use="required"/>
  <xs:attribute name="IdentityURL" type="xs:string" use="required"/>
  <xs:attribute name="IsDeviceAccount" type="BooleanType" use="required"/>
  <xs:attribute name="UserDeviceGuid" type="xs:string" use="required"/>
  <xs:attribute name="UserDeviceName" type="xs:string" use="required"/>
```

```
   <xs:attribute name="_EA1" type="xs:base64Binary" use="required"/>
   <xs:attribute name="_EA2" type="xs:base64Binary" use="required"/>
   <xs:attribute name="_EventID" type="xs:int" use="required"/>
   <xs:attribute name="created" type="xs:int" use="required"/>
  </xs:complexType>
  </xs:element>
  <xs:complexType name="FileUploadXlinkType">
  <xs:attribute name="Compress" type="BooleanType" use="required"/>
  <xs:attribute name="CompressMethod" type="xs:string" use="required"/>
  <xs:attribute name="MakeDBRelative" type="xs:boolean" use="required"/>
  <xs:attribute name="RenameHref" type="xs:boolean" use="required"/>
  <xs:attribute name="actuate" type="xs:string" use="required"/>
  <xs:attribute name="deserialize" type="xs:string" use="required"/>
  <xs:attribute name="href" type="xs:string" use="required"/>
  <xs:attribute name="role" type="xs:string" use="required"/>
  <xs:attribute name="serialize" type="xs:string" use="required"/>
  <xs:attribute name="show" type="xs:string" use="required"/>
  <xs:attribute name="title" type="xs:string" use="required"/>
  <xs:attribute name="link" type="xs:string" use="required"/>
   </xs:complexType>
 </xs:schema>
```

The "g:SE" element is specified in the **fragment** element schema in this section. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

The following table describes the payload XML elements and attributes:

XPath	Description
/fragment	Fragment element
/fragment/Event	Service event element
/fragment/Event/@BackupFragmentSize	Backup fragment size
/fragment/Event/@BackupFragmentCount	Backup fragment count
/fragment/Event/@BackupGUID	Backup GUID
/fragment/Event/@BackupIndexCount	Backup index count
/fragment/Event/@BackupSize	Backup size
/fragment/Event/@BackupVersion	Backup version, MUST be 3.
/fragment/Event/@DomainGuid	Domain GUID
/fragment/Event/@GUID	Account GUID
/fragment/Event/@GrooveVersion	Client version
/fragment/Event/@IdentityURL	Identity URL
/fragment/Event/@IsDeviceAccount	A Boolean value MUST be true for a device account
/fragment/Event/@UserDeviceGuid	Device GUID
/fragment/Event/@UserDeviceName	Client host name

XPath	Description
/fragment/Event/@_EA1	Base64 encoded file token data
/fragment/Event/@_EA2	Base64 encoded file data
/fragment/Event/@_EventID	Event identifier
/fragment/Event/@created	Message creation timestamp
/fragment/Event/SE	SE element
/fragment/Event/SE/Enc	Encrypted data element
/fragment/Event/SE/Auth	Authentication data element
/fragment/Event/SE/Auth/@MAC	Message Authentication Code
/fragment/Event/AuditElemUploadXlink	Link to log entry element
/fragment/Event/AuditElemUploadXlink/@Compress	MUST be 0.
/fragment/Event/AuditElemUploadXlink/@CompressMethod	MUST be "ZLIB"
/fragment/Event/AuditElemUploadXlink/@MakeDBRelative	MUST be "true"
/fragment/Event/AuditElemUploadXlink/@RenameHref	MUST be "false"
/fragment/Event/AuditElemUploadXlink/@actuate	MUST be "user"
/fragment/Event/AuditElemUploadXlink/@deserialize	MUST be "ignore"
/fragment/Event/AuditElemUploadXlink/@href	MUST be the URL of the file.
/fragment/Event/AuditElemUploadXlink/@role	Reserved. MUST be set to ""
/fragment/Event/AuditElemUploadXlink/@serialize	MUST be "ignore"
/fragment/Event/AuditElemUploadXlink/@show	MUST be "replace"
/fragment/Event/AuditElemUploadXlink/@title	File name
/fragment/Event/AuditElemUploadXlink/@link	MUST be "simple"

The **/fragment/Event/@_EA1** attribute contains data specified as:

```
<xs:element name="TK">
 <xs:complexType>
 <xs:attribute name="_body" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_iv" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_key" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_mac" type="xs:base64Binary" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the attributes:

XPath	Description
/TK	Token element
/TK/@_body	Token data
/TK/@_iv	Initialization vector for the encryption and decryption
/TK/@_key	Encrypted key
/TK/@_mac	Message Authentication Code

2.2.3.31 FileUploadResponse Message

The **FileUploadResponse** message is the non-fault response message for the **FileUpload** service request message.

```
<xs:element name="FileUploadResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.31.1 FileUploadResponse Payload

The **/FileUploadResponse/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="FUR">
 <xs:complexType>
 <xs:attribute name="_com" type="BooleanType" use="required"/>
 <xs:attribute name="_ha" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_sig" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_v" type="xs:string" use="required" fixed="1,0"/>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/FUR	Element contains file upload response element
/FUR/@_com	Indicates if an upload has finished
/FUR/@_ha	SHA1 hash of the unencrypted uploaded file
/FUR/@_sig	Signature
/FUR/@_v	Service version

2.2.3.32 FileUploadQuery Message

The **FileUploadQuery** message is a service request message.

```
<xs:element name="FileUploadQuery" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.32.1 FileUploadQuery Payload

The **/FileUploadQuery/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="FileUploadQuery">
 <xs:complexType>
 <xs:attribute name="_ha" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_v" type="xs:string" use="required" fixed="1,0"/>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/FileUploadQuery	Contains upload query element
/FileUploadQuery/@_ha	Hash
/FileUploadQuery/@_v	Service version

2.2.3.33 FileUploadQueryResponse Message

The **FileUploadQueryResponse** message is the non-fault response message for the **FileUploadQuery** service request message.

```
<xs:element name="FileUploadQueryResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.33.1 FileUploadQueryResponse Payload

The **/FileUploadQuery/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="FUQR">
 <xs:complexType>
 <xs:attribute name="_com" type="BooleanType" use="required"/>
 <xs:attribute name="_ha" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_sf" type="xs:int" use="required"/>
 <xs:attribute name="_sig" type="xs:base64Binary" use="required"/>
 <xs:attribute name="_v" type="xs:string" use="required" fixed="1,0"/>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/FUQR	Element contains file upload query response element.
/FUQR/@_com	Indicates if an upload has finished
/FUQR/@_ha	SHA1 hash of the unencrypted uploaded file
/FUQR/@_sf	File status code on the server: 0 = not present 1 = present 2 = processing
/FUQR/@_sig	Signature
/FUQR/@_v	Service version

2.2.3.34 IdentityPublish Message

The **IdentityPublish** message is a service request message.

```
<xs:element name="IdentityPublish" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.34.1 IdentityPublish Payload

The **/IdentityPublish/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="vCard">
   <xs:complexType>
   <xs:attribute name="Data" type="xs:base64Binary" use="required"/>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The following table describes the payload XML elements and attributes:

XPath	Description
/fragment/vCard	vCard element
/fragment/vCard/@data	vCard data

2.2.3.35 IdentityPublishResponse Message

The **IdentityPublishResponse** message is the non-fault response message for the **IdentityPublish** service request message.

```
<xs:element name="IdentityPublishResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.36 KeyActivation Message

The **KeyActivation** message is a service request message.

```
<xs:element name="KeyActivation" type="ServiceRequestType3"/>
```

The **ServiceRequestType3** is specified in section 2.2.2.2.18.

2.2.3.36.1 KeyActivation Payload

The **/KeyActivation/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="PayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **PayloadWrapper** element is specified in section 2.2.2.1.9.

The **/fragment/PayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="Payload">
 <xs:complexType>
 <xs:attribute name="GrooveVersion" type="xs:string"/>
 </xs:complexType>
</xs:element>
```

The following table describes the XML elements and attributes:

XPath	Description
/Payload	Payload element
/Payload/@GrooveVersion	Client version

2.2.3.37 KeyActivationResponse Message

The **KeyActivationResponse** message is the non-fault response message for the **KeyActivation** service request message.

```
<xs:element name="KeyActivationResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.37.1 KeyActivationResponse Payload

The **/KeyActivation/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
```

```
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** is specified in section 2.2.2.1.11.

The value of the **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="KeyActivation">
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **KeyActivation** element is specified in section 2.2.2.1.4.

2.2.3.38 ManagedObjectInstall Message

The **ManagedObjectInstall** message is a service request message.

```
<xs:element name="ManagedObjectInstall" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.38.1 ManagedObjectInstall Payload

The **/fragment/ManagedObjectInstall/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="ManagedObjectInstalled">
 <xs:complexType>
 <xs:attribute name="Domain" type="xs:string" use="required"/>
 <xs:attribute name="ID" type="xs:string" use="required"/>
 <xs:attribute name="IdentityURL" type="xs:string" use="required"/>
 <xs:attribute name="ServerURL" type="xs:string"/>
 <xs:attribute name="IssuedTime" type="xs:double"/>
 <xs:attribute name="Name" type="xs:string"/>
 <xs:attribute name="Type" type="xs:string" use="required"/>
 <xs:attribute name="UserNAME" type="xs:string" use="required"/>
 <xs:attribute name="VCard" type="xs:base64Binary"/>
 </xs:complexType>
```

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

The following table describes the elements and attributes:

XPath	Description
/fragment/ManagedObjectInstalled	A managed object installed element
/fragment/ManagedObjectInstalled/@Domain	Domain GUID of server
/fragment/ManagedObjectInstalled/@ID	Managed object identifier
/fragment/ManagedObjectInstalled/@ServerURL	Server URL
/fragment/ManagedObjectInstalled/@IssuedTime	Issued time
/fragment/ManagedObjectInstalled/@Name	Name of the managed object
/fragment/ManagedObjectInstalled/@Type	Type of the managed object
/fragment/ManagedObjectInstalled/@UserNAME	This is the host name of the device for a message originating from a device account. In all other cases, it will be the full name of the identity.
/fragment/ManagedObjectInstalled/@VCard	VCard data

2.2.3.39 ManagedObjectInstallResponse Message

The **ManagedObjectInstallResponse** message is the non-fault response message for the **ManagedObjectInstall** request message.

```
<xs:element name="ManagedObjectInstallResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.40 ManagedObjectStatus Message

The **ManagedObjectStatus** message is a service request message.

```
<xs:element name="ManagedObjectStatus" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.40.1 ManagedObjectStatus Payload

The **/fragment/ManagedObjectStatus/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<xs:element name="domainGUID">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="ManagedObject" minOccurs="1" maxOccurs="unbound">
    <xs:complexType>
     <xs:attribute name="ID" type="xs:string" use="required" />
     <xs:attribute name="IssuedTime" type=" xs:double" use="required" />
     <xs:attribute name="Name" type="xs:string" use="required" />
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="ConsistencyDigest" type="xs:string" use="required" />
  <xs:attribute name="ConsistencyDomainGUID" type="xs:string" use="required" />
  <xs:attribute name="ConsistencyIdentityURL" type="xs:string" use="required" />
  <xs:attribute name="DomainMember" type="xs:string" use="required" />
  <xs:attribute name="IdentityURL" type="xs:string" use="required" />
  <xs:attribute name="IsBeta" type="xs:string"/>
  <xs:attribute name="IsTrial" type="xs:string"/>
  <xs:attribute name="Name" type="xs:string" use="required" />
  <xs:attribute name="ProductID" type="xs:string" />
  <xs:attribute name="UserGUID" type="xs:string" use="required" />
  <xs:attribute name="UserName" type="xs:string" use="required" />
 </xs:complexType>
</xs:element>
</xs:schema>
```

The element name 'domainGUID' changes according to server domainGUID. The format of the name is the capital letter "D" followed by the domainGUID of the server. The **domainGUID** MUST be specified.

The following table describes the elements and attributes.

Attribute	Description
domainGUID	Domain GUID that identifies the management server.
domainGUID/ManagedObject	A managed object. A **domainGUID** can contain one or more managed objects.
domainGUID/ManagedObject/@ID	Managed object ID.
domainGUID/ManagedObject/@IssuedTime	Issued time in Coordinated Universal Time (UTC).
domainGUID/ManagedObject/@Name	Object name, specifies as a string.
domainGUID/@ConsistencyDigest	Base64 encoded digest. This digest is set by the client and is returned in the response message by the server. The client can set it to any valid Base64 value.

domainGUID/@ConsistencyDomainGUID	**Domain GUID, specified as a** string**.**
domainGUID/@ConsistencyIdentityURL	Identity URL, specified as a **string**.

domainGUID/@DomainMember	**Indicates a domain member (member machine), specified as an integer. The value MUST be "1" for domain members and zero ("0") for non-domain members.**
domainGUID/@IdentityURL	Identity URL, specified as a **string**.
domainGUID/@IsBeta	Indicates that the client is beta version client, specified as an integer. The value "1"indicates a beta client and zero ("0") indicates a non-beta client. This attribute is optional.
domainGUID/@IsTrial	Indicates that the client is a trial client, specified as an integer. The value "1" indicates a trial client and zero ("0") indicates a non-trial client. This attribute is optional.
domainGUID/@Name	Object name, specified as a **string**.
domainGUID/@ProductID	Product identifier, specified as a **string**. This attribute is optional. <2>
domainGUID/@UserGUID	User GUID, specified as a **string**.
domainGUID/@UserName	User name, specified as a **string**.

There can be multiple **ManagedObject** elements in the data.

2.2.3.41 ManagedObjectStatusResponse Message

The **ManagedObjectStatusResponse** message is the non-fault response message for the **ManagedObjectStatus** service request message.

```
<xs:element name="ManagedObjectStatusResponse" type="ServiceResponseType3"/>
```

The **ServiceResponseType3** is specified in section 2.2.2.2.21.

2.2.3.41.1 ManagedObjectStatusResponse Payload

The **/fragment/ManagedObjectStatusResponse/fragment/ManagedObjects/Payload/@data** attribute contains data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ManagedObjectsWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The referenced **ManagedObjectsWrapper** element is specified in the following schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="ManagedObjectsWrapper">
 <xs:complexType>
  <xs:sequence>
  <xs:element xmlns:g="urn:groove.net" ref="g:SE"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The "g:SE" element is specified in section 2.2.2.1.13. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

The **/fragment/fragment/ManagedObjectsWrapper/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="ManagedObjects">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="ManagedObject" type="ManagedObjectType"
      maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="ConsistencyDigest" type="xs:base64Binary"
     use="required"/>
 <xs:attribute name="ConsistencyDomainGUID" type="xs:string"
     use="required"/>
 <xs:attribute name="ConsistencyIdentityURL" type="xs:string"
     use="required"/>
 <xs:attribute name="IdentityURL" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

The **ManagedObjectType** is specified in section 2.2.2.2.10.

The following table describes the elements and attributes:

XPath	Description
/ManagedObjects	Managed objects element
/ManagedObjects/@ConsistencyDigest	Consistency digest
/ManagedObjects/@ConsistencyDomainGUID	Consistency domain GUID
/ManagedObjects/@ConsistencyIdentityURL	Consistency identity URL
/ManagedObjects/@IdentityURL	Identity URL
/ManagedObjects/ManagedObject	Managed object element

2.2.3.42 PassphraseResetRequest Message

The **PassphraseResetRequest** message is a service request message.

```
<xs:element name="PassphraseResetRequest" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.42.1 PassphraseResetRequest Payload

The **/PassphraseResetRequest/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The**/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="PassphraseResetRequest">
  </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:element name="Contact">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="vCard"/>
  <xs:element name="ClientDevices"/>
  <xs:element name="RelayDevices"/>
  <xs:element name="CSecurity" type="CSecurityType"/>
  </xs:sequence>
  <xs:attribute name="Flags" type="xs:int" use="required"/>
  <xs:attribute name="SeqNum" type="xs:int" use="required"/>
  <xs:attribute name="URL" type="xs:string" use="required"/>
  <xs:attribute name="Version" type="xs:string" use="required"
      fixed="0,0,0,0"/>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **CSecurityType** is specified in section 2.2.2.2.6. The **PassphraseResetRequest** element is specified in the following schema:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="urn:groove.net"/>
 <xs:element name="PassphraseResetRequest">
 <xs:complexType>
  <xs:sequence>
  <xs:element xmlns:g="urn:groove.net" ref="Contact"/>
  </xs:sequence>
  <xs:attribute name="AccountGUID" type="xs:string" use="required"/>
  <xs:attribute name="AdminPublicKeyHash" type="xs:base64Binary"
      use="required"/>
  <xs:attribute name="DigestAlgorithm" type="xs:string" use="required"/>
  <xs:attribute name="EncryptedMasterKey" type="xs:base64Binary"
      use="required"/>
```

```
  <xs:attribute name="EncryptedSecretMasterKey" type="xs:base64Binary"
      use="required"/>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The following table describes the elements and attributes:

XPath	Description
/fragment	Fragment element
/fragment/PassphraseResetRequest/@AccountGUID	Account GUID
/fragment/PassphraseResetRequest/@AdminPublicKeyHash	SHA1 hash of the domain's data recovery DER-encoded encryption public key.
/fragment/PassphraseResetRequest/@DigestAlgorithm	Digest algorithm
/fragment/PassphraseResetRequest/@EncryptedMasterKey	Encrypted master key
/fragment/PassphraseResetRequest/@EncryptedSecretMasterKey	Encrypted secret master key
/fragment/PassphraseResetRequest/Contact	Contact element
/fragment/PassphraseResetRequest/Contact/@Flags	The value MUST be 1.
/fragment/PassphraseResetRequest/Contact/@SeqNum	The value MUST be 1.
/fragment/PassphraseResetRequest/Contact/@URL	Identity URL
/fragment/PassphraseResetRequest/Contact/@Version	Contact version
/fragment/PassphraseResetRequest/Contact/vCard	vCard element
/fragment/PassphraseResetRequest/Contact/ClientDevices	ClientDevices element
/fragment/PassphraseResetRequest/Contact/RelayDevices	RelayDevices element
/fragment/PassphraseResetRequest/Contact/CSecurity	Security data element

2.2.3.43 PassphraseResetRequestResponse Message

The **PassphraseResetRequestResponse** message is the non-fault response message for the **PassphraseResetRequest** service request message.

```
<xs:element name="PassphraseResetRequestResponse"
            type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.44 PassphraseResetStatus Message

The **PassphraseResetStatus** message is a service request message.

```
<xs:element name="PassphraseResetStatus" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.44.1 PassphraseResetStatus Payload

The **/PassphraseResetStatus/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="PassphraseResetStatus">
 <xs:complexType>
 <xs:attribute name="AccountGUID" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
```

2.2.3.45 PassphraseResetStatusResponse Message

The **PassphraseResetStatusResponse** message is the non-fault response message for the **PassphraseResetStatus** service request message.

```
<xs:element name="PassphraseResetStatusResponse" type="ServiceResponseType2"/>
```

The **ServiceResponseType2** is specified in section 2.2.2.2.20.

2.2.3.45.1 PassphraseResetStatusResponse Payload

The **/PassphraseResetStatusResponse/Payload/@data** attribute contains the payload data specified as:

```
<xs:schema xmlns:g="urn:groove.net" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:groove.net"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="fragment">
 <xs:complexType>
  <xs:sequence>
  <xs:element ref="ReturnPayloadWrapper"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

The **ReturnPayloadWrapper** element is specified in section 2.2.2.1.11.

The **/fragment/ReturnPayloadWrapper/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="ReturnPayload">
 <xs:attribute name="Data" type="xs:base64Binary" use="required"/>
</xs:element>
```

The **Data** can be decoded as UTF8 to an XML string. The schema of the decoded XML string is:

```
<xs:element name="PassphraseResetStatus">
 <xs:complexType>
 <xs:attribute name="AccountGUID" type="xs:string" use="required"/>
 <xs:attribute name="EncryptedMasterKey" type="xs:base64Binary"/>
 <xs:attribute name="EncryptedSecretMasterKey" type="xs:base64Binary"/>
 <xs:attribute name="Status" type="xs:int" use="required"/>
 </xs:complexType>
</xs:element>
```

The following table describes the attributes:

XPath	Description
/PassphraseResetStatus	Passphrase reset status element
/PassphraseResetStatus/@AccountGUID	Account GUID
/PassphraseResetStatus/@EncryptedMasterKey	Encrypted master key
/PassphraseResetStatus/@EncryptedSecretMasterKey	Encrypted secret master key
/PassphraseResetStatus/@Status	MUST be one of the following values 0: Success 1: Failure 3: Poll again for an update

2.2.3.46 StatisticsPackage Message

The **StatisticsPackage** message is a service request message.

```
<xs:element name="StatisticsPackage" type="ServiceRequestType1"/>
```

The **ServiceRequestType1** is specified in section 2.2.2.2.16.

2.2.3.46.1 StatisticsPackage Payload

The **/StatisticsPackage/Payload** element contains the **fragment** data element specified in section 2.2.2.1.3.

The **/fragment/Event/SE/Enc/@EC** attribute data before encryption is specified as:

```
<xs:element name="StatisticsPackage">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="Stat" maxOccurs="unbounded">
  <xs:complexType>
   <xs:attribute name="AccountGUID" type="xs:string" use="required"/>
   <xs:attribute name="IdentityURL" type="xs:string" use="required"/>
   <xs:attribute name="Long" type="xs:int" use="required"/>
   <xs:attribute name="Long2" type="xs:int" use="required"/>
   <xs:attribute name="String" type="xs:string" use="required"/>
   <xs:attribute name="String2" type="xs:string" use="required"/>
```

```
    <xs:attribute name="Type" type="xs:int" use="required"/>
   </xs:complexType>
   </xs:element>
  </xs:sequence>
  </xs:complexType>
 </xs:element>
```

The following table describes the elements and attributes:

XPath	Description
/StatisticsPackage	Statistics package element
/StatisticsPackage/Stat	Statistics data element
/StatisticsPackage/Stat/@AccountGUID	Account GUID
/StatisticsPackage/Stat/@IdentityURL	Identity URL
/StatisticsPackage/Stat/@Long	This element has a different use depending on the value of the Type element. Elapsed time for statistic types 0,15,19,20 Active members for statistic type 25
/StatisticsPackage/Stat/@Long2	This element has a different use depending on the value of the Type element. Visit count for statistic type 0,15 User count for statistic type 19,20 Total members for statistic type 25
/StatisticsPackage/Stat/@String	This element has a different use depending on the value of the Type element. Tool template URL for statistic type 0, 16,17,19,20,21,24,25 Canonical Telespace URL for statistic type 15
/StatisticsPackage/Stat/@String2	This element has a different use depending on the value of the Type element. Canonical Telespace URL for statistic type 0,16 Telespace name for statistic type 15,17,24
/StatisticsPackage/Stat/@Type	MUST be one of the following values: 0: Tool time 15: Telespace time 16: Tool create 17: Telespace create 19: Voice over IP 20: Voice over IP – conference mode 21: Telespace delete 24: Telespace join 25: Telespace members

2.2.3.47 StatisticsPackageResponse Message

The **StatisticsPackageResponse** message is the non-fault response message for the **StatisticsPackage** service request message.

```
<xs:element name="StatisticsPackageResponse" type="ServiceResponseType1"/>
```

The **ServiceResponseType1** is specified in section 2.2.2.2.19.

2.2.3.48 GMSConfig Message

The client sends (HTTP GET) an empty request to http://<server>/GMSConfig server URL to retrieve authenticated and non-authenticated server URLs.

2.2.3.49 GMSConfig Response

The server responds to GMSConfig request by adding the following header elements to the response.

ServerVersion: MUST contain a numeric value representing major version of the server<3>.

NormalProtocol: MUST be "http://" or "https://".

NormalPath: MUST contain non-authenticated server path enclosed between "/"

AuthProtocol: MUST be "http://" or "https://".

AuthPath: MUST contain authenticated server path enclosed between "/".

3 Protocol Details

3.1 Common Details

3.1.1 Modified Alleged RC4

This protocol uses a variation of the RC4 algorithm called **Modified Alleged Rivest Cipher 4 (MARC4) algorithm**.

RC4 algorithm is specified in [SCHNEIER].

MARC4 is different from RC4 as following:

- Both encryption and decryption MUST use the initialization vector (IV), and IV MUST be the same size as the secret key.
- A new random IV MUST be generated every time that the data is encrypted. A matching IV MUST be used every time that the data is decrypted.
- The IV MUST be XOR-ed with the secret key, and the result MUST be used as the secret key for RC4.
- The first 256 bytes of the key stream MUST be discarded. Subsequent bytes of the key stream MUST be used in the same manner as they are used with RC4.

3.1.2 ASN.1 Syntax for DER-encoding Public Key used for Encryption

The Diffie-Hellman public key used for **ElGamal encryption** MUST be DER-encoded as follows (using ASN.1 syntax):

```
DHPublicKeyForElGamalEncryption::= SEQUENCE {
      p     INTEGER, -- prime, p
      q     INTEGER OPTIONAL, -- factor of p-1, only present when p = j*q+1,
                              -- where j is not 2
      g     INTEGER, -- generator, g
      y     INTEGER -- public key (g^x mod p, where x is the private key)
}
```

Diffie-Hellman is a cryptographic key exchange protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret over an insecure communications channel, as specified in [CRYPTO].

Rivest, Shamir and Adleman (RSA) is a cryptographic protocol as specified in [CRYPTO].

The RSA public key used for RSA encryption MUST be DER-encoded as follows (using ASN.1 syntax):

```
RSAPublicKeyForRSAEncryption::= SEQUENCE {
      modulus           INTEGER, -- the RSA modulus n
      publicExponent    INTEGER  -- the RSA public exponent e
}
```

3.1.3 Management Server Certificate

A certificate to participate in this protocol MUST be in X.509.v3 format as defined in [RFC3280]. It MUST also have the following **object identifier (OID) (2)** extensions defined:

Extension OID	Description
2.16.840.1.114227.1.1.1	Encryption public key. The value MUST be an RSA 2048-bit encryption public key, DER encoded.
2.16.840.1.114227.1.1.2	Encryption public key algorithm. The value MUST be an RSA **Unicode** string without the terminating NULL character.
2.16.840.1.114227.1.1.3	Encryption algorithm. The value MUST be an RSA Unicode string without the terminating NULL character.

These certificates are self-signed, with the issuer and subject being the same, and validity set to 100 years.

The following is a list of fields and values for a sample domain certificate:

```
X.509 Certificate:
Version: 3
Serial Number: 54b91b4c1294bfceb4919e76feb1ef6747291d7a
Signature Algorithm:
  Algorithm ObjectId: 1.2.840.113549.1.1.5 sha1RSA
  Algorithm Parameters:
  05 00
Issuer:
  OU=ems5
  O=ems5

NotBefore: 1/2/2008 11:58 AM
NotAfter: 1/2/2108 11:58 AM

Subject:
  OU=ems5
  O=ems5

Public Key Algorithm:
  Algorithm ObjectId: 1.2.840.113549.1.1.1 RSA
  Algorithm Parameters:
  05 00
Public Key Length: 2048 bits
Public Key: UnusedBits = 0
  0000 30 82 01 08 02 82 01 01 00 c8 94 24 8b 06 09 c2
  0010 26 95 b7 ae 0c af 7d 8a c0 9e de df f1 1b 42 48
  0020 e2 59 c1 9b 81 ad c2 f6 eb b9 61 ca 6a 8e 96 45
  0030 1b 1a 99 6c f0 d9 0a 42 71 61 73 cd 17 e5 a8 16
  0040 f1 fb b0 e0 99 90 e4 9c b3 a9 b1 05 21 73 fd f3
  0050 fa 16 60 2f c5 4e 42 16 02 78 96 5f 0c 55 2a c4
  0060 10 4f 80 fd 92 49 b7 53 67 52 66 75 49 48 a2 3b
  0070 c9 75 67 34 7c c5 7b 41 27 71 ca ab e3 ea 2e f9
  0080 b4 c3 08 ce 36 40 cd a3 a0 31 98 dd 11 ef e4 d0
  0090 16 10 7a 6c 15 81 a7 39 19 1a 73 10 e9 b5 b4 99
  00a0 bf 43 ce d8 82 d6 c5 9e 8c 9b b6 39 08 e4 89 37
  00b0 b4 a2 c5 cd c2 fa b8 74 d5 91 d8 6d 72 be e4 51
  00c0 34 7d 70 06 54 0c c1 82 7c 63 34 3b 43 5d d6 1d
```

```
    00d0 dd 72 79 fb fc 8d fb 75 da eb 89 59 25 4d e8 59
    00e0 aa 40 cb 1f 7b 8e 65 61 2c c3 e1 5b 95 e0 83 30
    00f0 70 f1 a7 ae dd 75 16 08 c0 ab b0 90 8e 98 0d 31
    0100 48 89 ba 1c 3d 64 de 5a f5 02 01 11
  Certificate Extensions: 3
    2.16.840.1.114227.1.1.1: Flags = 0, Length = 10c
    Unknown Extension type

    0000 30 82 01 08 02 82 01 01 00 da 83 6a 2c 61 3b 37  0..........j,a;7
    0010 06 50 f1 a6 58 9c 78 47 4f 65 ea 53 a0 78 e9 3a  .P..X.xGOe.S.x.:
    0020 50 c5 2a 32 cd 02 a9 59 94 dd 65 a9 ea 50 fd 42  P.*2...Y..e..P.B
    0030 b9 d5 a2 c5 de 3c 97 51 0a d1 db ed 83 bf a2 d1  .....<.Q........
    0040 66 67 0e 41 ff be 15 aa b6 1a 8e f3 5d 28 4f 3c  fg.A........](O<
    0050 8f 01 1f 21 38 6b 83 14 2d 38 94 62 3c 86 50 ce  ...!8k..-8.b<.P.
    0060 99 bc 6c d1 fa 14 21 1d bb b9 97 84 ef db 80 31  ..l...!........1
    0070 cc dc 56 c5 a4 13 ca 11 15 50 32 52 e3 c3 0d e3  ..V......P2R....
    0080 e6 a1 b4 b3 96 2e 53 70 5f e0 0a 1d 3d 29 5b 53  ......Sp_...=)[S
    0090 03 ca 32 8c ca 00 47 83 b2 52 8c ea aa 69 bd dc  ..2...G..R...i..
    00a0 f3 e9 f4 43 84 79 b0 b4 0f b9 1d 37 bc d3 d7 7e  ...C.y.....7...~
    00b0 3d 98 7f 0e f3 34 4b d7 1f 75 97 e8 06 36 27 fd  =....4K..u...6'.
    00c0 74 33 b5 7c c4 fc 0f 56 9b ae 95 04 42 0d e2 7a  t3.|...V....B..z
    00d0 bd 7c 26 44 71 10 6a 5e 51 ad 4b 55 0f 17 4e 48  .|&Dq.j^Q.KU..NH
    00e0 d7 b8 0c 0d 94 a6 26 3e d5 b0 f3 92 89 00 ef 58  ......&>.......X
    00f0 4a cd 11 9c 4a 2f 4c 1b 26 17 cb e3 ec 91 7c 6f  J...J/L.&.....|o
    0100 bc d2 55 67 eb 61 64 0a ed 02 01 11        ..Ug.ad.....

    2.16.840.1.114227.1.1.2: Flags = 0, Length = 6
    Unknown Extension type

    0000 52 00 53 00 41 00                R.S.A.

    2.16.840.1.114227.1.1.3: Flags = 0, Length = 6
    Unknown Extension type

    0000 52 00 53 00 41 00                R.S.A.

  Signature Algorithm:
    Algorithm ObjectId: 1.2.840.113549.1.1.5 sha1RSA
    Algorithm Parameters:
    05 00
  Signature: UnusedBits=0
    0000 97 68 e4 f4 c8 36 ef 9f 09 8d 41 28 42 7b 71 49
    0010 34 a8 31 fd 1b 3c 6a 33 2f 86 98 f3 02 e8 48 b5
    0020 7c cf e3 98 d2 e6 ed 5d 6b 80 76 39 0e 35 4a da
    0030 4c f3 fa 39 02 04 b4 14 40 66 2f 84 fb 14 36 b8
    0040 03 f6 b5 6b 40 6f ee 37 6d f6 99 04 2c 24 e6 44
    0050 87 1e 76 11 30 cd 14 58 33 11 7b d8 69 3a 23 3b
    0060 f4 3f c3 4f 13 f9 e4 52 26 de 51 cd 37 9e f4 ce
    0070 f0 0a e0 af 5c f9 71 1a 84 d2 cf 05 d3 cf 44 bd
    0080 57 3d 18 dd 27 0e b4 85 4d 3f 35 c6 e7 6e 63 47
    0090 02 1a 1d 83 6b c0 69 55 94 5d 67 2a 82 8f 20 cf
    00a0 32 42 52 1e b7 2f 13 62 3e 24 6f a2 4c d5 c1 9f
    00b0 4d 3f 14 aa d8 54 46 13 b0 08 dc e7 dd ae b9 72
    00c0 20 cf 6a 01 a0 e1 04 b5 d0 96 7b f4 04 87 c9 47
    00d0 66 c6 e1 df 3c c6 1b e2 49 53 51 12 2c 77 b4 02
    00e0 f1 cc c2 52 d2 02 2f 26 fa ce f3 91 6f 8d 61 c9
    00f0 11 13 a3 ea 52 0e 93 f7 0f b6 d4 9a 1d 15 5a 9e
  Signature matches Public Key
  Root Certificate: Subject matches Issuer
```

```
Key Id Hash(sha1): 49 1f 36 d0 24 e1 2a 05 e1 af a9 14 ce 31 9c 98 53 04 53 a2
Cert Hash(md5): 98 14 cd 87 ac 14 d0 8f a8 8f 0b 2b 3d 61 b1 bc
Cert Hash(sha1): cb 55 5b 26 a5 98 e5 ca 22 d0 45 fc 26 3f cb 5f 91 f2 95 27
```

3.1.4 Security Model for Audit-Related Methods

The domain's key pair is used to encrypt *(RSA, 2048-bit)* and sign *(RSA signature, 2048-bit)* communications between the client and the management server. For each account and management domain pair on the client, the client generates a secret (symmetric) key *(MARC4, 192-bit)* when it first communicates with the management server. The client sends the secret key to the management server encrypted with the encryption public key from the domain's certificate. All subsequent client-to-management-server communications are encrypted with this secret key. The management server signs all responses it sends to clients using the signature key from the domain's private key file. The client verifies the signature using the corresponding public key from the domain's certificate.

The management domain's key pair *(RSA, 2048-bit)* is used to encrypt all audit log data. For performance reasons, the audit log is encrypted with a per-session secret (symmetric) key (**Advanced Encryption Standard (AES)**, 192-bit) that is in turn encrypted with the encryption key from the management domain's certificate. The encryption is done using the AES CTR mode. The management server uses the management domain's encryption private key to decrypt each log session's secret key and then uses that key to decrypt the log. The encryption public key from the management domain's certificate is also used to encrypt tokens for encrypted binary files and the Audit Service secret key (see following).

The Audit Service secret key (AES, 192-bit) is used to lock-on a client to a management server. When the Audit Service first runs on a device it generates this secret (symmetric) key, which it stores in a file that can only be accessed by administrators. The service transmits this key, encrypted with the management server key, along with the **device URL** with every log upload. The management server, at log decryption time, tests to see if that Audit Service secret key is already associated with a device URL. If it does exist and matches the current pair then the decryption proceeds. If it does not match, an error is generated. If it doesn't exist, the management server adds the pair (that is locks-on). The Audit Service secret key is also used to MAC the log to prevent rogue devices from uploading logs purported to be from a different device.

To encrypt the audit log, when a new client **session** begins, the Audit Service generates a new symmetric encryption key (AES, 192-bit) for the session. Next the Audit Service encrypts the symmetric with the encryption key from the management server's public key and writes the encrypted **symmetric key** in the log in a "begin session" entry. The Audit Service saves the unencrypted symmetric key in memory and uses it to encrypt the subsequent log entries.

Before every upload, the ranges of the entries in the log that are to be uploaded are MAC'ed using the Audit Service secret key. MAC'ing the log file allows the management server to verify that separate uploads originates from the same device. The Audit Service secret key is transmitted to the management server via a lock-on message.

3.1.5 Common Security Processing for Securing a Payload with a Shared Key

This section specifies the common security processing rules for securing a payload with a shared key between a client and a server. This is used by a server to secure any response that carries payload, and by a client to secure requests.

3.1.5.1 Inputs

Header element: a secured fragment element as defined in sections under 2.2.3 for request or response payload. As the input, this element MUST have the name "fragment" in the namespace "urn:groove.net". It MUST have one child as the payload wrapper element. The name for this payload wrapper element is the name of the service. The payload wrapper element MUST contain one child element named "SE" in the namespace "urn:groove.net". The "urn:groove.net:SE" element is referred to as the security element. It MUST contain no content.

Payload element: application request or response element. This element contains content to be secured.

Shared Key: key used for encryption and data-integrity protection. This is the key shared by a client and a server. It can be a key derived from an account configuration code or a key established in a **CreateAccount** message. Unless explicitly stated otherwise in this protocol, the shared secret key used is established in a **CreateAccount** message. Cases where the key is derived from an account configuration code are explicitly specified.

3.1.5.2 Serialize Header Element

Header element MUST be serialized into a byte representation. As part of the serialization, all attributes for each element MUST be sorted alphabetically, with no compression and no extra white spaces. **UTF-8** encoding MUST be used. In addition, the following rules MUST be followed to ensure successful security processing:

- The following MUST be present at the beginning of the serialized data:

```
<?xml version='1.0'?><?groove.net version='1.0'?>
```

- The top level **fragment** element (in the namespace "urn:groove.net") MUST use the namespace prefix "g", with "xmlns" attribute defining "g" as "urn:groove.net".
- The second level **PayloadWrapper** element MUST NOT have any namespace prefix.
- The third level **SE** element (in the namespace "urn:groove.net") and its sub-elements (all in the namespace "urn:groove.net") MUST use the namespace prefix "g" without any "xmlns" attribute.

3.1.5.3 Serialize Payload Element

The payload element MUST be serialized into a byte representation. As part of the serialization, all attributes for each element MUST be sorted alphabetically, with no compression and no extra white spaces. UTF8 encoding MUST be used. In addition, the following rules MUST be followed to ensure successful security processing:

- The following MUST be present at the beginning of the serialized data:

```
<?xml version='1.0'?><?groove.net version='1.0'?>
```

- A namespace prefix or "xmlns" attribute MUST NOT be added.

3.1.5.4 Compute Message Digest

The message digest MUST be computed using the SHA1 algorithm, defined in [RFC3174].

The message digest MUST include the following in the specified order: serialized header element (as defined in section 3.1.5.2), and the serialized payload element (as defined in section 3.1.5.3).

3.1.5.5 Encrypt Serialized Payload Element

A byte representation of the serialized payload element MUST be encrypted using the MARC4 algorithm (as defined in section 3.1.1), with the shared key.

A unique initialization vector MUST be generated for each message, and used during encryption of that message.

3.1.5.6 Compute Message Authentication Code

The Message Authentication Code MUST be computed using the **Hash-based Message Authentication Code (HMAC)** -SHA1 algorithm (as defined in [RFC4634]), with the shared key.

The Message Authentication Code MUST be for the message digest, as defined in section 3.1.5.4.

3.1.5.7 Create Encrypted Element

The encrypted element named "Enc" in the namespace "urn:groove.net" MUST be created as a child of the security element "g:SE", as defined in section 2.2.2.2.22. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

An encrypted element MUST have the attributes as defined in section 2.2.2.2.22. Specifically:

- The **EC** attribute MUST have the encrypted serialized payload element, as defined in section 3.1.5.5.
- The **IV** attribute MUST be the initialization vector for the encrypted serialized payload element, as defined in section 3.1.5.5.

3.1.5.8 Create Authenticator Element

An authenticator element named "Auth" in the namespace "urn:groove.net" MUST be created as a child of the security element "g:SE", as defined in section 2.2.2.2.22. The SE element MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

The authenticator element MUST have the attributes defined in section 2.2.2.2.22. Specifically:

- The **MAC** attribute MUST be the Message Authentication Code, as defined in section 3.1.5.6.

3.1.5.9 Serialize Secured Fragment into Output

After the preceding procedures have been performed, the header element is the secured **fragment** element.

The secured **fragment** element, containing the security element, MUST be serialized into a byte representation. The same serialization rules specified in section 3.1.5.2 MUST be applied here.

The serialized secured **fragment** is the output.

3.1.6 Common Security Processing for Opening Secured Content with a Shared Key

This section specifies the common security processing rules for opening a payload with content secured with a shared key between a client and a server. These rules are used by a server for opening most of the secured requests from a client, and by a client for opening secured responses from a server.

3.1.6.1 Inputs

Secured fragment element: a secured fragment element as defined in sections under section 2.2.3 for request or response payload. As the input, this element MUST have the name of "fragment" in the namespace "urn:groove.net". It MUST have one child element as the payload wrapper element. The payload wrapper element MUST contain one child element named "SE" in the namespace "urn:groove.net". "urn:groove.net:SE" element MUST be as defined in section 2.2.2.2.22, with all sub-elements and attributes. The "urn:groove.net:SE" element is referred as the security element.

Shared Key: a key used for decryption and data-integrity verification. This is the key shared by a client and a server. It can be a key derived from an account configuration code, or a key established via a **CreateAccount** message. Unless explicitly stated otherwise in this protocol, the shared secret key used is established in a **CreateAccount** message. Cases in this protocol where the key is derived from an account configuration code are explicitly specified.

3.1.6.2 Parse Encrypted Element

The encrypted element "urn:groove.net:Enc" MUST be a child of the security element "urn:groove.net:SE" and MUST have the attributes as defined in section 2.2.2.2.22.

The following attributes MUST be parsed and saved as inputs into decryption:

- The **EC** attribute is the encrypted serialized payload element.
- The **IV** attribute is the initialization vector for the encrypted serialized payload element.

3.1.6.3 Parse Authenticator Element

The authenticator element "urn:groove.net:Auth" MUST be a child of the security element and MUST have the attributes defined in section 2.2.2.2.22.

The following attributes MUST be parsed and saved as inputs into data integrity verification:

- The **MAC** attribute is the Message Authentication Code.

3.1.6.4 Delete Encrypted Element and Authenticator Element

The encrypted element "urn:groove.net:Enc" and authenticator element "urn:groove.net:Auth" MUST be deleted as child objects of the security element, which is part of the input secured fragment element.

Once this is done, the secured **fragment** element becomes a header element, as defined in 3.1.5.1.

3.1.6.5 Serialize Header Element

The header element MUST be serialized as defined in section 3.1.5.2.

3.1.6.6 Decrypt Serialized Payload Element

The encrypted, serialized payload element MUST be decrypted using the MARC4 algorithm (as defined in section 3.1.1), with the shared key.

A corresponding per-message initialization vector from the encrypted element MUST be used, as defined in section 3.1.6.2.

3.1.6.7 Compute Message Digest

The message digest MUST be computed using the SHA1 algorithm, as defined in [RFC3174]

The message digest MUST include the following in the specified order: serialized header element (as defined in section 3.1.6.5, and the decrypted content (the result from section 3.1.6.6).

3.1.6.8 Verify Data Integrity

The Message Authentication Code MUST be computed using the HMAC-SHA1 algorithm (as defined in [RFC4634]), with the shared key.

The Message Authentication Code MUST be for the message digest, as computed in section 3.1.6.7.

Comparison of this Message Authentication Code with the one saved from section 3.1.6.3 MUST be performed. If the Message Authentication Codes do not match, the data integrity verification fails. The message MUST be rejected.

3.1.6.9 Deserialize Decrypted Content into Output

The serialized payload element, which is the decrypted content from section 3.1.6.6, MUST be deserialized from a byte representation into an XML representation, as the payload element.

This payload element is the output.

3.2 Server Details

3.2.1 Abstract Data Model

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

- **Management Server:** The management server is an entity representing the management server installation. The management server has the following attribute:
  - **Server URL:** MUST be a string representing the HTTP address of the management server. The address MUST use http://*hostname*/gms.dll syntax, where *hostname* is the name of the management server in a domain name form (such as contoso.com).<4>
- **Management Domains:** A collection of entries corresponding to management domains defined on the management server. A domain GUID uniquely identifies each entry. Each entry includes the following attributes:
  - **Domain Name:** A string containing name of the domain. The name is unique among all domains on the management server.

- **Enterprise PKI:** A Boolean value that is set to false for member contacts to be signed by the management server.
- **Certification authority (CA) Name:** The name to be represented in the domain certificate.
- **Domain Certificate:** A X.509.v3 certificate as defined in section 3.1.3, containing two sets of 2048-bit RSA public keys, one for encryption and one for signing.
- **Data Recovery Certificate:** A X.509.v3 certificate as defined in section 3.1.3, containing two sets of 2048-bit RSA public keys, one for encryption and one for signing.

- **Accounts:** A collection of entries corresponding to the client accounts. An account can either be associated with a device or with a user. The account GUID in conjunction with the domain GUID uniquely identifies an entry in this collection. Each entry includes the following attributes:
  - **Account GUID:** This GUID is the same as the account GUID on the client.
  - **Domain GUID:** In conjunction with the account GUID, uniquely identifies an account entry.
  - **Secret Key**: A 192-bit symmetric secret key used for integrity protecting and encrypting messages between the client and the management server.
- **Device Accounts**: A collection of entries corresponding to client device accounts. Device GUID MUST uniquely identify an entry in this collection. Each entry includes the following attributes:
  - **Domain GUID:** A value representing the domain to which the device belongs.
  - **Device GUID:** This GUID is the Account GUID for the device and is in the accounts table.
  - **DPG GUID:** An optional value representing the device policy group entry provisioned to the device account.
  - **Status:** Represents the management status of the device. The value is one of the following:
    - 0 = Not Managed (default)
    - 1 = Managed
    - -1 = Deleted
- **Members**: A collection of entries representing the members managed on the management server. A member GUID uniquely identifies each member. The configuration code and configuration code hash is unique for each entry. Each entry includes the following attributes:
  - **Member GUID**: A unique GUID used to identify the member.
  - **Pre-Authentication Token**: A unique identifier for the member on this management server; could be the same as the member GUID.
  - **Full Name:** Represents full name of the member.
  - **E-mail Address:** A value representing the e-mail address of the member.
  - **Account Configuration Code:** A unique GUID used for binding the member to a client identity.
  - **KeyID:** The SHA1 hash of the SHA1 hash of the account configuration code. For computing the SHA1 hash, the account configuration string is treated as a Unicode string. The SHA1 hash is

computed by interpreting the string as bytes, in **little-endian** order, not including the terminating NULL character. The SHA1 hash is first performed on the account configuration code and then the SHA1 hash is applied to the results of the first SHA1 hash.

- **First Name:** An optional value that specifies the member's first name.
- **Last Name:** An optional value that specifies member's last name.
- **Organization:** An optional value that specifies the member's organization.
- **Title:** An optional value that specifies the member's organization title.
- **ORG Street 1:** An optional value that specifies line 1 of a member's organization street.
- **ORG Street 2:** An optional value that specifies line 2 of a member's organization street.
- **ORG City:** An optional value that specifies the member's organization city name.
- **ORG State:** An optional value that specifies the member's organization state name.
- **ORG Postal Code:** An optional value that specifies the member's organization postal code.
- **ORG Phone:** An optional value that specifies the member's organization phone number.
- **ORG Cell:** An optional value that specifies the member's organization cell phone number.
- **ORG Fax:** An optional value that specifies the member's organization fax number.
- **Remote User:** An optional value that specifies the member's login name.
- **Status:** Represents the state of a member. The value is one of the following
  - 1: Pending (default)
  - 2: Active
  - 3: Disabled
  - -1: Deleted
  - -2: Migrated
- **Identity URL:** A unique identifier issued to an identity by the client.
- **Account GUID:** A unique identifier issued by the client to identify an account. In conjunction with Identity URL, uniquely identifies a member.
- **IPG GUID:** A GUID representing the identity policy group provisioned to member. If the value is null on creation, then this column inherits the domain's default identity policy group GUID.
- **RSG GUID:** A GUID representing the relay server group provisioned to the member. If the value is null on creation, then this column inherits the domain's default relay server group GUID.
- **Contact URL:** Same as the identity URL. Set during the enrollment process.
- **Contact Security:** Contains contact security data.
- **MigrationStatus:** A Boolean value that is set to true to migrate the member to another domain or server.

- **Migration Server:** Contains the URL of the management server to which the member is to be migrated, if migration status is set to true.

- **Affiliation:** Contains a string representing the member's affiliation. See section 3.2.5.1.5.

- **Relay Server Sets:** A collection of entries corresponding to the relay server sets available on the server. Each relay server set can contain one or more relay servers. A GUID uniquely identifies each entry in the collection. Each entry can include the following attribute:

  - **RSG GUID:** A unique GUID used to identify a relay server set.

- **Identity Policy Templates:** A collection of entries, each entry referencing a unique set of identity policy managed objects. A GUID uniquely identifies each entry in the collection. Each entry can include the following attribute:

  - **IPG GUID:** A unique GUID used to identify an identity policy template.

- **Device Policy Templates**: A collection of entries referencing a set of device policy managed objects. A GUID uniquely identifies each entry in the collection. Each entry can include the following attribute:

  - **DPG GUID:** A unique GUID used to identify a device policy template.

- **Relay Servers:** A collection of entries corresponding to one or more relay servers registered with the management server. A GUID uniquely identifies each entry. A relay server entry can contain following attributes:

  - **RS GUID**: A unique GUID used to identify the Relay Server.

  - **RSG GUID**: A unique GUID MUST identify the Relay Server Set to which this relay server belongs.

  - **RSG Sequence**: A number defining the ordering of this relay server within the relay server set.

  - **Device URL:** Contains a string representing a relay device URL. The address uses "grooveDNS://hostname" syntax, where hostname is the name of the server in domain name format (such as fabrikam.com). The client uses the device URL to contact the relay server.

  - **SOAP URL:** Contains a string representing the HTTP address of the relay server. The address uses http://hostname/ syntax, where hostname is the name of the server in domain name format (such as fabrikam.com). Management server to contact relay server uses the SOAP URL.

  - **SSTP Certificate:** An X.509.v3 certificate, containing two sets of asymmetric key pairs, one for encryption and one for signing. Used for securing the registration message between client and relay server. See [MS-GRVSPMR] for details.

  - **SOAP Certificate**: An X.509.v3 certificate, containing two sets of asymmetric key pairs, one for encryption and one for signing. Used for securing the registration message between management server and relay server. See [MS-GRVSPMR] for details.

- **Account Backups**: A collection of entries representing client account data. An account backup can be spread across one or more account backup entries. The backup GUID binds all account backup entries belonging to the same account. The backup GUID in conjunction with the fragment index uniquely identifies an account backup entry. Each entry can include the following attributes:

  - **Account GUID**: GUID of the account to which the account backup belongs.

- **Backup GUID**: A unique GUID used in conjunction with the fragment index to identify an account backup entry.
- **Fragment Index**: Represents the sequence number of the account backup fragment.
- **Fragment Count**: Represents the number of fragments in the account backup.
- **Backup Data**: Contains the account backup fragment.

- **Passphrase Reset Requests:** A collection of short-lived entries, representing manual passphrase reset request data. An account GUID uniquely identifies a passphrase reset request entry. Each entry can include the following attributes:
  - **Account GUID:** An account GUID to bind the request to one or more Members.
  - **Digest Algorithm:** This value is "SHA1".
  - **Admin Public Key Hash:** This value is the SHA1 hash of the domain's DER-encoded encryption public key.
  - **Encrypted Master Key:** This is a client-generated secret key encrypted in the domain's encryption public key.
  - **Encrypted Secret Master Key:** This is a client-generated secret key encrypted in the domain's encryption public key.
  - **Temporary Contact:** Contact element from the request. This attribute is used for generating the identity cipher key used to obtain an encryption public key used to re-encrypt the storage master key before sending it to the client as part of the manual password reset transaction.
  - **Status:** Represents the state of the PassphraseResetRequest. The valid states are:
    - 0: Request is approved
    - 1: Request is not approved
    - 3: Request is pending
  - **Password**: Password to be used for securing a manual passphrase reset request.
  - **Server Encrypted Master Key:** This is a management server encrypted master key using the encryption public key from the temporary contact sent by the client.
  - **Server Encrypted Secret Master Key:** This is a management server encrypted secret master key using the encryption public key from the temporary contact sent by the client.
- **Managed Objects**: A collection of entries corresponding to the managed objects available on the management server. An Object GUID uniquely identifies a managed object. Each entry contains the following attributes:
  - **Object GUID:** A GUID used to identify the managed object. If the managed object is an identity template managed object, then this value is the same as a member GUID in the Members collection.
  - **PG GUID:** An optional policy group GUID to bind the managed object to an identity or device policy group.
  - **Object Type:** An integer value representing the type of managed object contained in the entry.

- **Object Template:** A Unicode string that contains the object template.
- **Object Data:** An optional base64-encoded string containing the managed object data.
- **IssuedTime:** The time at which the object data column was updated or created, represented in milliseconds since midnight 01/01/1970.

- **Audit Chunk Data Storage**:
  - **GUID:** A GUID to identify fragments belonging to the same audit event data.
  - **Account GUID:** The GUID of the account sending this data.
  - **Version:** The version of the client.
  - **Index Count:** Represents the sequence number of the current data fragment.
  - **Fragment Count:** The number of fragments in this event data.
  - **Fragment Size:** Size of the fragment.
  - **Total Size:** Total size of the event data.
- **Audit Devices:**
  - **Device Guid:** A unique GUID to identify the auditing device.
  - **Device GUID Hash:** This element contains the SHA1 hash of the device GUID. For computing SHA1 hash, the device GUID string is treated as a Unicode string. The SHA1 hash is computed by interpreting the string as bytes, in little-endian order, not including the terminating NULL character.
  - **Device Lock On Key:** A secret key shared between the client device and the management server for securing audit-related messages.
  - **Last Purge Sequence:** An integer representing the sequence number of the last audit log event from this device.
- **Audit File Storage:**
  - **Hash:** A unique string representing the SHA1 hash of the file content.
  - **Hash Algo:** The name of the algorithm used for computing the digest.
  - **Status:** A representation of the current state of the file upload:
    - 0: Not present
    - 1: Present
    - 2: Processing
  - **Length:** Length of the file
  - **File Data:** File content

3.2.2 Timers

None.

3.2.3 Initialization

The server MUST expose the web service path "gms.dll" off the root URL.

The server MUST create and expose the service path "/AutoActivate/gms.dll" to enable Auto Account configuration. The "AutoActivate/gms.dll" endpoint MUST <5>

1. Support HTTP and HTTPS protocols.

2. Support an authentication scheme.

The Server MUST have a management domain with at least one member.

For every management domain, the server MUST create a certificate as defined in section 3.1.3.

To enable clients to activate their accounts, the server SHOULD create members in management domains. Each new member MUST have an account configuration code and a KeyID is derived by and SHA1 hash of an SHA1 hash of the account configuration code. For computing the SHA1 hash, the account configuration string is treated as a Unicode string. The SHA1 hash is computed by interpreting the string as bytes, in little-endian order, not including the terminating NULL character.

If the server is to support auto account configuration or auto activation, the server SHOULD associate members with their login names.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

The following diagram is a high-level sequence diagram that illustrates the operation of the protocol:

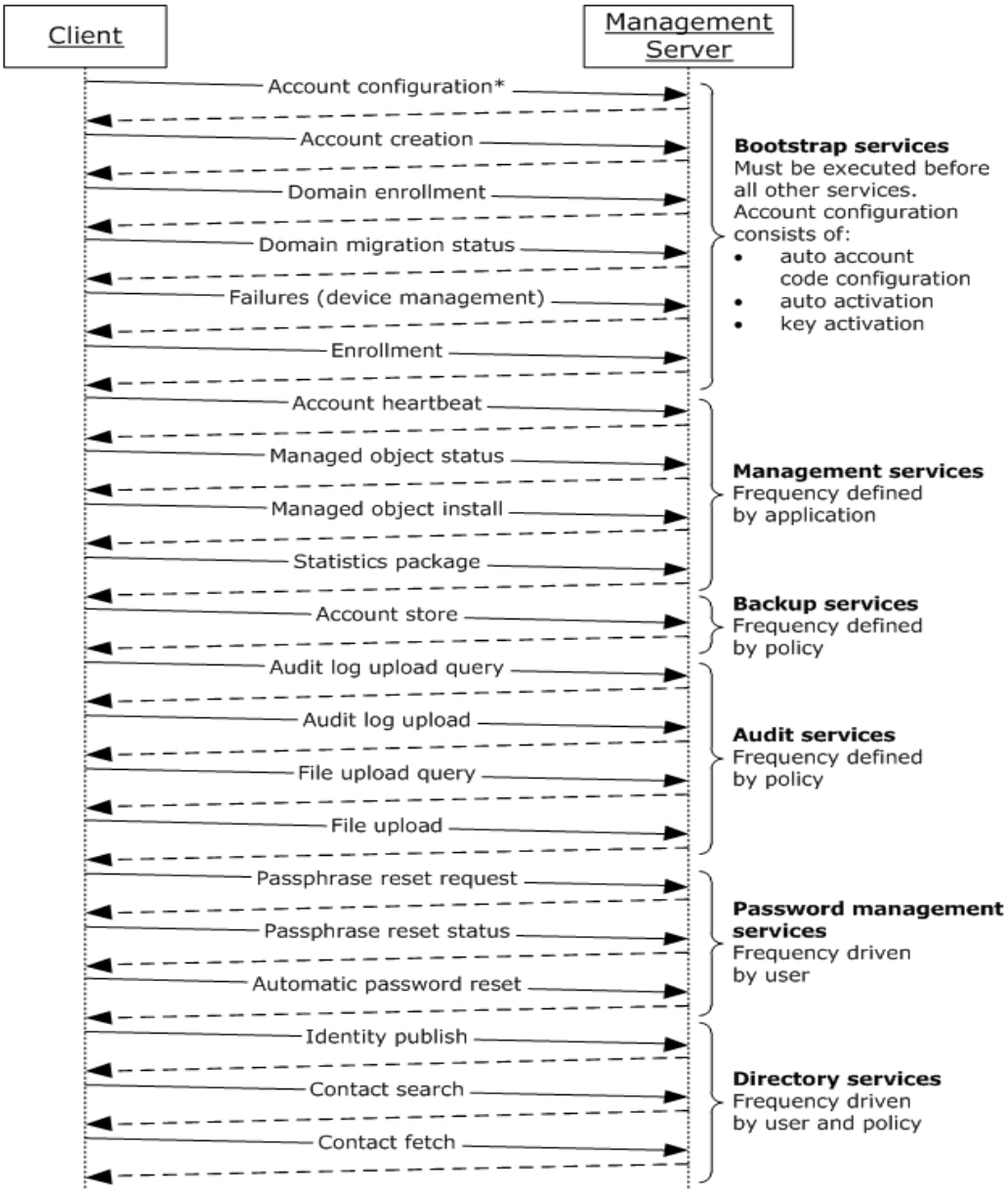

Figure 4: Message sequence between protocol client and protocol server

3.2.5.1 Common Processing Rules

3.2.5.1.1 Common Security Processing Rules for Messages Secured with Key Derived from Account Configuration Code

This section specifies the common security processing rules for opening a request secured with a shared key derived from an account configuration code, and then securing the response with the same key.

This processing applies to the **KeyActivation** and **DomainEnrollment** messages.

3.2.5.1.1.1 Request Parsing

The server MUST follow the specific request definition in section 2.2.3 to parse the envelope. This includes base64-decoding the **PayloadWrapper** element.

If the parsing fails, or any required element or attribute is missing, a SOAP fault response MUST be returned. The fault code is defined in Section 2.2.2.2.15 with the value set to either "105", "203", or "204". The fault string MAY be any string describing the failure.

3.2.5.1.1.2 Retrieve Matching Key

The **PayloadWrapper** element is secured with a key that is derived from the account configuration code for the client. The **KeyID** attribute from the "urn:groove.net:SE" element carries the digest of the key. The **KeyID** value serves as the index for retrieving the corresponding key for the account configuration code.

The server MUST extract the fragment element as defined under section 2.2.3 for the specific message. This **fragment** element contains the secured content.

Using the **KeyID** value, the server MUST search its database to find the corresponding account configuration code and the key. If no code is found, the server MUST return a SOAP fault with fault code set to "401". The fault string MAY be any string describing the failure.

3.2.5.1.1.3 Open Secure Content

With the **fragment** element and the key derived from the account configuration code, the server MUST successfully open the secured content using the shared secret key to satisfy the security requirement. If the server cannot open the secured content, it ignores the message. For details, see section 3.1.6.

3.2.5.1.1.4 Process and Prepare Response

The server MUST process the request (specified in the specific section for the message) and prepare a response as defined in Section 2.2.3 to prepare the response payload.

3.2.5.1.1.5 Secure Response

After following these procedures, the response payload is the payload element defined in section 3.1.5.1.

The server MUST also prepare a fragment element with "fragment" name in the namespace "urn:groove.net". The **fragment** element MUST have one child payload wrapper element. The payload wrapper element name and attributes are specific to the message. The server MUST set the attributes for this payload wrapper element as defined by this protocol for the specific message. The payload wrapper element MUST have one child element named "SE" in the namespace "urn:groove.net". This "urn:groove.net:SE" element MUST contain no content and is referred to as the security element. After following these procedures, this **fragment** element is the header element as defined in section 3.1.5.1.

With the **fragment** element, the response payload, and the shared key, the server MUST follow the steps in section 3.1.5 to secure the response and restore the serialized secured fragment element.

3.2.5.1.1.6 Package Payload into Response Envelope and Send

The server MUST package the secured fragment into a response envelope as defined under section 2.2.3. The server then serializes the envelope and returns it to the client.

3.2.5.1.2 Common Security Processing Rules for Messages Secured with Shared Account Key

This section specifies the common security processing rules for opening a request envelope secured with an account key shared between a client and a server.

This processing applies to all messages except for the **AutoAccountCodeConfiguration**, **CreateAccount**, **DomainEnrollment**, and **KeyActivation** messages.

3.2.5.1.2.1 Request Parsing and Key Retrieval

Upon receiving a request, the server MUST parse the message as defined in Section 2.2.3. If any element or attribute is missing, the server MUST return a **Fault**. The fault code is as defined in section 2.2.2.2.15, with the value set to either "105", "203" or "204". The fault string MAY be any string describing the failure.

The server MUST extract the **fragment** element as defined under section 2.2.3 for the specific message. This **fragment** element contains the secured content.

The server MUST open the **GUID** and **DomainGUID** attributes from the **Event** element. These are the account GUID (this can be a device account or a user account) and the domain GUID. Using these two GUIDs, the server MUST query its database to retrieve the shared key for the account and the domain.

If the server fails to find the account or the shared key, the server MUST return a **Fault**. The fault code MUST be set to "200". The fault string MAY be any string describing the failure.

3.2.5.1.2.2 Open Secured Content

The server MUST follow rules specified in section 3.1.6 with the **fragment** element and the shared key as inputs, and open the secured content to restore the request payload element.

3.2.5.1.2.3 Process and Prepare Response

The server processes the request and prepares the response as defined under section 2.2.3.

3.2.5.1.2.4 Secure Response

If the response only has a return code, the response message is not secured. All messages that return **ServerResponseType1** as specified in 2.2.2.2.19 have a response that only has a return code. The server MUST continue without securing the return code to package the payload into a response envelope and send to the client as specified in section 3.2.5.1.2.5.

If the response carries payload, the response payload MUST be secured.

The response payload becomes the payload element as defined in section 3.1.5.1.

The server MUST also prepare a **fragment** element with the name "fragment" in the namespace "urn:groove.net". The **fragment** element MUST have one child payload wrapper element. The payload wrapper element name and attributes are specific to the message. The server MUST set the attributes for this payload wrapper element as defined by the message definition. The payload

wrapper element MUST have one child element named "SE" in the namespace "urn:groove.net". This "urn:groove.net:SE" element MUST contain no content and is referred to as the security element. This **fragment** element becomes the header element as defined in section 3.1.5.1.

With the **fragment** element, the response payload, and the shared key, the server MUST follow the steps in section 3.1.5 to secure the response and restore the serialized secured fragment element.

3.2.5.1.2.5 Package Payload into Response Envelope And Send

The server MUST package the secured fragment or the unsecured response code into a response envelope as defined under section 2.2.3. The server then serializes the envelope and returns it to the client.

3.2.5.1.3 Populating the Data Model

The management server implementation MUST bootstrap the data model with the following information for it to participate in this protocol.

Management Server

There MUST be only one entry in the management server collection. The server URL attribute MUST contain a valid value.

Management Domains

There MUST be at least one entry in the management domains collection for the management server to participate in this protocol.

Identity Policy Templates

There MUST be at least one identity policy template per management domain. An identity policy template MUST create the following managed objects.

1. Identity Policy
2. Domain Trust Policy
3. Data Recovery Policy

Device Policy Templates

There MUST be at least one device policy template per management domain. A device policy template MUST create the following managed objects.

1. Device Policy
2. Account Services Policy
3. Data Recovery Policy
4. Passphrase Policy
5. Component Update Policy

Relay Server Sets

There MUST be at least one relay server set per management domain. A relay server set MUST reference zero or more relay servers.

Relay Servers

The relay server registration file can be used for populating a relay server entry. The following unqualified XPATHs table provides the relationship between the relay server column and xml attributes.

Column	XPATH
Device URL	/RelayAttributes/@RelayDeviceURL
SOAP URL	/RelayAttributes/@SOAPURL
SOAP Certificate	/RelayAttributes/@SOAPCertificate
SSTP Certificate	/RelayAttributes/@SSTPCertificate

A relay server MUST be assigned to a relay server set to be provisioned to a member. For more details, see [MS-GRVSPMR].

Members

There MUST be at least one member in a domain for the management server to participate in this protocol.

The member GUID, full name, e-mail, configuration code, and configuration code hash attributes MUST contain valid values.

The remote user column MUST match the REMOTE_USER (authenticated user name) value returned by the web server to participate in auto activation or auto account configuration methods.

A member MUST have a valid identity policy group GUID and relay server group GUID. If these attributes are null, the member MUST inherit the values from its parent domain's identity policy group GUID and relay server group GUID.

An active Member MUST have valid identity URL, account GUID, contact security, and contact URL attributes. The contact column MUST contain the published contact.

Accounts

Each entry in the accounts collection represents an account-domain combination. The account GUID in conjunction with the domain GUID MUST uniquely identify each entry in the accounts table. A successful **CreateAccount** request creates an entry in this table.

Device Accounts

Each entry in this table represents a client device. An entry in this collection MUST exist for every device account. On creation, the entry MUST inherit the domain's device policy group GUID.

Account Backups

Each row in this collection represents an account backup fragment. The **AccountStore** method provides the data for populating this collection.

Passphrase Reset Requests

The Passphrase Reset Requests collection contains manual passphrase reset request information. The **PassphraseResetRequest** method provides the data for populating this table.

Managed Objects

There MUST be one entry in this collection for each object managed by the management server.

Creation of a new Member MUST result in creation of an identity template managed object in the managed objects table. The object GUID MUST be the same as the member GUID.

Creation of a new policy group MUST result in creation of associated policy group managed objects. Each newly created managed object MUST refer to its parent policy group in the PC GUID column. Any changes to the object template MUST result in an updated object data and a new issued time.

An identity template managed object MUST construct its object data based on member information. Any changes to member information or relay server provisioning MUST result in an updated object data.

All managed object elements MUST be prefixed with "g:" where "g" stands for the namespace "xmlns:g="urn:groove.net".

3.2.5.1.4 Generating Managed Object Data

This section describes the process for populating the object data attribute of a managed object entry. The object data contains a base64-encoded string representing a serialized managed object. Section 2.2.2.2.10 defines the schema for various managed objects used in this protocol. A managed object consists of an object header as defined in section 2.2.2.2.12, a body which varies with each managed object type, and a signature part as defined in section 2.2.2.2.13. The process for populating managed object data has three steps.

Step 1: Populate Managed Object Header

The managed object header contains information about the object and the domain managing the object.

Section 2.2.2.2.9 defines the schema for the management domain information contained in the header. The management domain information is identical for all managed objects in that domain. The content for the management domain information references the management domain and the management server entry specified in the abstract data model.

Section 2.2.2.2.12 defines the schema for the managed object header. The attribute values differ for each managed object as defined in the following tables:

Account Services Policy

XPath	Description
/ManagedObjectHeaderType/@Name	MUST contain value "grooveAccountServicesPolicy2:"
/ManagedObjectHeaderType/@DisplayName	MUST be value "Account Services Policy"
/ManagedObjectHeaderType/@Description	Same as display name
/ManagedObjectHeaderType/@GUID	MUST be the managed object GUID
/ManagedObjectHeaderType/@IntendedIdentityURL	MUST be empty
/ManagedObjectHeaderType/@IssuedTime	MUST be managed object issued time in double, represented as milliseconds since midnight 01/01/1970

XPath	Description
/ManagedObjectHeaderType/@ReplacementPolicy	MUST be value "$IssuedTime"

Component Update Policy

XPath	Description
/ManagedObjectHeaderType/@Name	MUST be value "grooveDeviceBehavior://ComponentUpdatePolicy"
/ManagedObjectHeaderType/@DisplayName	MUST be value "Groove Update Policy"
/ManagedObjectHeaderType/@Description	Same as display name
/ManagedObjectHeaderType/@GUID	MUST be the managed object GUID
/ManagedObjectHeaderType/@IntendedIdentityURL	MUST be empty
/ManagedObjectHeaderType/@IssuedTime	MUST be managed object issued time in double, represented as milliseconds since midnight 01/01/1970
/ManagedObjectHeaderType/@ReplacementPolicy	MUST be value "$IssuedTime"

Data Recovery Policy

XPath	Description
/ManagedObjectHeaderType/@Name	MUST be value "grooveAccountPolicy2://DataRecovery"
/ManagedObjectHeaderType/@DisplayName	MUST be value "Groove Data Recovery Policy"
/ManagedObjectHeaderType/@Description	Same as display name
/ManagedObjectHeaderType/@GUID	MUST be the managed object GUID
/ManagedObjectHeaderType/@IntendedIdentityURL	MUST be empty
/ManagedObjectHeaderType/@IssuedTime	MUST be managed object issued time in double, represented as milliseconds since midnight 01/01/1970
/ManagedObjectHeaderType/@ReplacementPolicy	MUST be value "$IssuedTime"

Device Policy

XPath	Description
/ManagedObjectHeaderType/@Name	MUST be value "grooveDevicePolicy:"
/ManagedObjectHeaderType/@DisplayName	MUST be value "Device Policy"
/ManagedObjectHeaderType/@Description	Same as display name
/ManagedObjectHeaderType/@GUID	MUST be the managed object GUID

XPath	Description
/ManagedObjectHeaderType/@IntendedIdentityURL	MUST be empty
/ManagedObjectHeaderType/@IssuedTime	MUST be managed object issued time in double, represented as milliseconds since midnight 01/01/1970
/ManagedObjectHeaderType/@ReplacementPolicy	MUST be value "$IssuedTime"

Domain Trust Policy Object

XPath	Description
/ManagedObjectHeaderType/@Name	MUST be value "grooveDomainTrustPolicy://*DomainGUID*/*ObjectGUID*" Where ▪ *DomainGUID* is the management domain GUID ▪ *ObjectGUID* is the managed object GUID
/ManagedObjectHeaderType/@DisplayName	MUST be value "Domain Trust Policy"
/ManagedObjectHeaderType/@Description	Same as display name
/ManagedObjectHeaderType/@GUID	MUST be the managed object GUID
/ManagedObjectHeaderType/@IntendedIdentityURL	MUST be empty
/ManagedObjectHeaderType/@IssuedTime	MUST be managed object issued time in double, represented as milliseconds since midnight 01/01/1970
/ManagedObjectHeaderType/@ReplacementPolicy	MUST be value "$IssuedTime"

Identity Object

XPath	Description
/ManagedObjectHeaderType/@Name	MUST be the value. "grooveIdentity://*MemberGUID*" Where *MemberGUID* is the GUID of the member referenced by this managed object
/ManagedObjectHeaderType/@DisplayName	MUST be the full name of the member
/ManagedObjectHeaderType/@Description	MUST be the value "Groove Identity"
/ManagedObjectHeaderType/@GUID	MUST be the managed object GUID
/ManagedObjectHeaderType/@IntendedIdentityURL	MUST be empty
/ManagedObjectHeaderType/@IssuedTime	MUST be managed object issued time in double, represented as milliseconds since midnight 01/01/1970
/ManagedObjectHeaderType/@ReplacementPolicy	MUST be the value "$Always"

Identity Policy Object

XPath	Description
/ManagedObjectHeaderType/@Name	MUST be the value. "grooveIdentityPolicy2:"
/ManagedObjectHeaderType/@DisplayName	MUST be the value "Identity Policy"
/ManagedObjectHeaderType/@Description	MUST be the value "Identity Policy"
/ManagedObjectHeaderType/@GUID	MUST be the managed object GUID
/ManagedObjectHeaderType/@IntendedIdentityURL	MUST be empty
/ManagedObjectHeaderType/@IssuedTime	MUST be managed object issued time in double, represented as milliseconds since midnight 01/01/1970
/ManagedObjectHeaderType/@ReplacementPolicy	MUST be the value "$IssuedTime"

PassphrasePolicy

XPath	Description
/ManagedObjectHeaderType/@Name	MUST be the value. "groovePassphrasePolicy2:"
/ManagedObjectHeaderType/@DisplayName	MUST be the value "Passphrase Policy"
/ManagedObjectHeaderType/@Description	MUST be the value "Passphrase Policy"
/ManagedObjectHeaderType/@GUID	MUST be the managed object GUID
/ManagedObjectHeaderType/@IntendedIdentityURL	MUST be empty
/ManagedObjectHeaderType/@IssuedTime	MUST be managed object issued time in double, represented as milliseconds since midnight 01/01/1970
/ManagedObjectHeaderType/@ReplacementPolicy	MUST be the value "$IssuedTime"

Step 2: Populate Managed Object Body

There are two types of managed objects: a policy object and an identity object. A management server administrator sets the content for a policy object, while a member entry defined in the abstract data model provides the data for an identity object.

Managed objects are stored in managed object collections as defined in the abstract data model. Each entry in the collection contains object data representing the managed object. Any changes to a policy or a member attribute MUST result in an updated managed object data.

Account Services Policy

Section 2.2.2.2.10.1 defines the schema for this managed object.

Component Update Policy

Section 2.2.2.2.10.2 defines the schema for this managed object.

If the component update policy is set to "Allow", the client is allowed for component updates over the Internet. If the policy is set to "Deny", the client is not allowed for component updates. If the policy is set to "Local", the client is allowed for component updates on the local system.

Data Recovery Policy

Section 2.2.2.2.10.3 defines the schema for this managed object. Any changes to the policy attribute or management domain's data recovery certificate MUST result in an updated managed object data.

Device Policy

Section 2.2.2.2.10.4 defines the schema for this managed object.

Domain Trust Policy

Section 2.2.2.2.10.5 defines the schema for this managed object. Each policy MUST contain one item to establish a cross-domain trust relationship. Multiple domain trust policies MUST be used to establish cross-domain trust relationships with more than one domain. Any changes to the management domain's name or certified authority name MUST result in an updated managed object data.

Identity

Section 2.2.2.2.10.6 defines the schema for this managed object. This managed object represents a member entry as defined in the abstract data model. The data for generating this managed object is provided by the management domain, members, relay server sets, and relay server entries specified in the abstract data model. Any changes to the member attributes or relay server provisioning MUST result in an updated managed object data. Instructions for populating the identity template are given in the following table:

XPath	Description
/fragment/ManagedObject/Body/IdentityTemplate	Identity template element.
/fragment/ManagedObject/Body/IdentityTemplate/@Flags	MUST be one of the following values: 1: Valid member if the member status is active or pending. 3: Disabled member if the member status is disabled.
/fragment/ManagedObject/Body/IdentityTemplate/ManagementDomainMigration	This element MUST be present if and only if the member's migration status is true .
/fragment/ManagedObject/Body/IdentityTemplate/ManagementDomainMigration/@ServerURL	MUST be member's migration server URL .
/fragment/ManagedObject/Body/Contact	Contact element containing signed information about the member VCARD, relay and presence server

XPath	Description
	assignments, and management domain information.
/fragment/ManagedObject/Body/Contact/VCard	vCard element.

/fragment/ManagedObject/Body/Contact/VCard/@Data	This base64 encoded UTF-8 string contains the member's VCARD data in the following format:
	
	`BEGIN:VCARD`
	`VERSION:2.1`
	`CS:UTF-8`
	`FN: [[- FN -]]`
	`N:[[- N -]]`
	`EMAIL;PREF;INTERNET:[[- EMAIL -]]`
	`TITLE:[[- ORG Title -]]`
	`ORG:[[- ORG -]]`
	`ADR;POSTAL;WORK:[[- ORG ADDRESS -]]`
	`TEL;WORK;VOICE:[[- ORG PHONE NUMBER -]]`
	`TEL;PAGER:[[- ORG CELL PHONE NUMBER -]]`
	`TEL;WORK;FAX:[[- ORG FAX NUMBER -]]`
	`END:VCARD`
	
	[[- FN –]]: Member's full name.
	[[- Email -]]: E-mail address of the member.
	[[- N -]] : If first name and last name are present, this field MUST be created by concatenating the first name, comma, and last name. If only the last name is present, this field MUST represent the last name.
	[[- ORG TITLE -]]: Member's title.
	[[- ORG -]]: Name of the organization the member belongs to.
	[[- ORG ADDRESS-]]: MUST be created by concatenating the member's Org Street 1, comma, Org Street 2, comma, Org City,

XPath	Description
	comma, Org State, comma, Org Postal Code, comma, and Org Country. [[- ORG PHONE NUMBER -]]: Member's business phone number. [[- ORG CELL PHONE NUMBER -]]: Member's business cell phone number. [[- ORG FAX NUMBER -]]: Member's business fax number. VCARD is extensible by client; server will only process the preceding fields.
/fragment/ManagedObject/Body/Contact/RelayDevices	This element and sub elements represent relay servers provisioned to the member. Member's relay server set provides the information for this element.
/fragment/ManagedObject/Body/Contact/RelayDevices/RelayDevice	There MUST be a relay device element for each relay server associated with member's relay server set. The sequence number (RSG Sequence) of the relay server governs the order of relay devices.
/fragment/ManagedObject/Body/Contact/RelayDevices/RelayDevice/@AuthorizationToken	MUST be the member's Pre-authentication token.
/fragment/ManagedObject/Body/Contact/RelayDevices/RelayDevice/@Certificate	MUST be the Relay Server's SSTP certificate
/fragment/ManagedObject/Body/Contact/RelayDevices/RelayDevice/@URL	MUST be the Relay Server's device URL.
/fragment/ManagedObject/Body/Contact/PresenceDevices	This element and sub elements represent relay servers provisioned to the member. The member's relay server set provides the information for this element. PresenceDevices and RelayDevices MUST specify the same devices in the same order. Each

XPath	Description
	element of these lists MUST be the identical URL, Certificate, and AuthorizationToken as the corresponding element of the other list.
/fragment/ManagedObject/Body/Contact/PresenceDevices/PresenceDevice	There MUST a presence server device element for each relay server associated with the member's relay server set. The sequence number (RSG Sequence) of the relay server governs the order of presence server devices.
/fragment/ManagedObject/Body/Contact/PresenceDevices/PresenceDevice/@URL	MUST be the Relay Server's device URL.
/fragment/ManagedObject/Body/Contact/PresenceDevices/PresenceDevice/@Certificate	MUST be the Relay Server's SSTP certificate.
/fragment/ManagedObject/Body/Contact/PresenceDevices/PresenceDevice/@AuthorizationToken	MUST be the member's Pre-authentication token.
/fragment/ManagedObject/Body/Contact/CustomFields	This element MUST only be present for a member who has finished the enrollment process.
/fragment/ManagedObject/Body/Contact/CustomFields/@_95_95Affiliation	MUST be the member's affiliation attribute as defined in section 3.2.5.1.5.
/fragment/ManagedObject/Body/Contact/CustomFields/@_95_95_95Affiliation_95Flags	This value MUST be 0x4000000.
/fragment/ManagedObject/Body/Origin	This element MUST be present for a member who has finished the enrollment process. The element data is copied from the member's contact security.
/fragment/ManagedObject/Body/Origin/ManagementDomain	This element MUST be present for a member who has finished the enrollment process. The element data is copied from the member's contact security.
/fragment/ManagedObject/Body/Contact/Certificate	This element MUST be present for a member who has finished the

XPath	Description
	enrollment process and the management domain is not using Enterprise PKI.
/fragment/ManagedObject/Body/Contact/Certificate/@ExpirationDate	MUST be the certificate expiration date in milliseconds since midnight 01/01/1970.
/fragment/ManagedObject/Body/Contact/Certificate/@SignerAddress	MUST be the server URL of the management server.
/fragment/ManagedObject/Body/Contact/Certificate/@SignerKeyHash	The value MUST be the SHA1 hash of the DER-encoded signature public key from the domain certificate.
/fragment/ManagedObject/Body/Contact/Certificate/@Signature	The management server MUST implement the following rules to generate the signature: ▪ The contact element MUST only contain the following elements: ▪ vCard element ▪ CustomFields element ▪ Origin element ▪ ManagementDomain element ▪ Certificate element ▪ Serialize contact element as XML, with UTF-8 encoding and sorted attributes. ▪ Compute the SHA1 hash of the serialized contact element. ▪ Generate the signature by signing the SHA1 hash with the signature private key from the domain's certificate using the RSA algorithm.

Identity Policy

Section 2.2.2.2.10.7 defines the schema for this managed object.

Passphrase Policy

Section 2.2.2.2.10.8 defines the schema for this managed object. The management server MUST implement the following rules for constructing the "vector" attribute value:

- The value MUST be a comma-separated list of integers.
- Each integer's string representation MUST be between 1 and 9 characters long.
- The final integer MUST not have anything after it (such as no comma after the final integer).
- The client uses the number of failed logon attempts as an index into this array of integers. Negative elements of this array are ignored for this purpose.
- Integer values MUST be positive, nonzero, and monotonically increase.
- Each value specifies the delay in seconds that the client waits before allowing the next logon attempt after a failed attempt.
- If the number of failed logon attempts exceeds the size of the vector, the last positive entry is used.
- Negative integer values are allowed as follows:
  - Each negative value MUST occur no more than once.
  - A value of -1 specifies that the client is to lockout the account (once the number of failed logon attempts is greater than the index of this value). If the value -1 is present in a list, it MUST be the last value in the list.
  - A value of -3 specifies that the client SHOULD display a message about delay because of failed logon attempts (once the number of failed logon attempts is greater than the index of this value).
  - All other negative values are reserved and MUST not be used.

Step 3: Securing the Managed Object

Section 2.2.2.2.13 defines the content of the security element. Instructions for populating the security element are given in the following table:

XPath	Description
/ObjectSignatureType/Signature	This element contains the signature of the managed object.
/ObjectSignatureType/Signature/@Fingerprint	This value is reserved and MUST be set to zero.
/ObjectSignatureType/Signature/@Value	To compute this value, the management server MUST implement the following rules: 1. Delete "urn:groove.net:Signatures" element from the managed object. 2. Serialize the managed object as XML, with UTF-

XPath	Description
	8 encoding and sorted attributes. 3. Compute the SHA1 hash of the preceding. 4. Sign the hash computed in the preceding step with domain's signature private key. 5. Set this attribute to the signed hash.

3.2.5.1.5 Generating a Member Affiliation Attribute

The management server MUST implement the following process for generating the member affiliation field:

- Fetch Unicode strings containing the member's domain name and member name.
- For each name string:
  - The Unicode string is converted into UTF-8 as defined in [RFC2279]
  - The UTF-8 value is hex-encoded:
    - Each byte is converted into the hexadecimal representation.
    - Each of the two resulting hexadecimal digits is then converted into the ASCII character ('0'-'9' and 'a'-'f').
    - The comma character ',' is added as a separator between each value (two hexadecimal digits representing a byte), but is not added after the last value.
  - The comma-separated string is then converted into a Unicode string.
  - The string "{<2.5.4.11=[13]" is pre-pended and the string ">}" is appended to the hex-encoded value.
- One Unicode string is generated for each name. To form the result, these Unicode strings are concatenated, using "/" as a separator. The separator MUST not be used if there is only one string. It also MUST not be added after the last string.
- When setting the attribute value, replace non-valid XML characters such as '<' and '>' with valid strings such as "<" and ">"

The newly created string is stored in the member's affiliation attribute. Any changes to the member's management domain name or full name MUST result in an updated affiliation attribute.

3.2.5.1.6 Manual Password Reset

On change of password reset status from pending to approved, the management server MUST implement the following rules to decrypt and re-encrypt master keys:

1. MUST use the management domain's data encryption private key to decrypt encrypted storage master keys information using the RSA algorithm as defined in [PKCS1].
2. MUST take the first 20 bytes of the result, and save it as a verifier for the following comparison.

3. MUST take the remaining bytes of the result, and save it as a master key itself for following processing.

4. MUST compute the SHA1 hash of the account GUID and the master key, in this order.

   - AccountGUID is a Unicode string, which, for the purposes of the SHA1 hash, MUST be interpreted as a byte array, not including the NULL-terminator (two bytes).

5. MUST compare the verifier saved in rule 2 with the result of the hash.

6. If not equal, the management server MUST send a Fault message with a 218 fault code to the client when the client asks for the status of reset.

7. Otherwise, the management server MUST encrypt the master key using the encryption public key from the temporary contact sent by the client.

8. The encryption public key MUST be parsed as defined in section 3.1.2.

9. Encryption MUST be done using RSA or ElGamal encryption.

10. Both newly encrypted master keys MUST be persisted until it is sent to the client.

The master key is used for encrypting the account keys and the secret master key is used to establish the handshake communication. See section 5 for a description of the master key and secret master key.

3.2.5.2 Bootstrap Services

A set of services to bootstrap entities involved in this protocol.

3.2.5.2.1 Account Configuration and Creation Services

Account configuration methods enable a management server to associate a client identity with a member. Client deployment infrastructure dictates the type of account configuration method used.

3.2.5.2.1.1 KeyActivation

Synopsis:

This method provides a mechanism for the management server to associate a client **Identity** with a management server **member** based on a **configuration code hash**. A client calls this method based on a configuration code and management server URL entered by an end-user or based on the configuration code provided in the response to an auto activation request by the management server.

Request Validation:

Parameter	Validation conditions	Fault code if validation fails
KeyID	MUST be valid and present on the server	401: Activation code invalid
KeyID	MUST NOT belong to a deleted or disabled member	401: Activation code invalid
KeyID	MUST NOT belong to an active member	402: Activation code already enrolled

Data Processing:

The data processing specified in this section references members and managed objects tables specified in the abstract data model.

The management server MUST locate the member using KeyID as the key. The management server MUST return identity template and policy managed objects associated with the member.

Results:

If no faults occur during the operation, the management server MUST return a **KeyActivationResponse** message as defined in section 2.2.3.36.1. In case of error it MUST return one of the fault codes defined in the following table.

Validation Check	Fault code if validation fails
Invalid KeyID	401: Activation code invalid
KeyID belongs to an active member.	402: Activation code already enrolled

3.2.5.2.1.2 AutoAccountCodeConfiguration

Synopsis:

This method provides a mechanism for the management server to bind a client **identity** with a management server **member** using a server side authentication mechanism. This method SHOULD<6> be received on an authenticated, **Secure Sockets Layer (SSL)** enabled port.

Request Validation:

Parameter / Web server	Validation conditions	Fault code if validation fails
AUTH_TYPE	The value MUST NOT be empty	227: Auto configuration directory is not secured
Authenticated user.	Authenticated member MUST be identifiable in management server.	401:Remote user " REMOTE_USER " not found
Authenticated user	Authenticated member MUST belong to a pending or active state.	401: Activation code invalid
Authenticated user and IsDomainMigration	If IsDomainMigration has a true value, then the authenticated member status MUST NOT be active	402: Activation code already enrolled
Authenticated user	If the status of authenticated member is active, then an account backup MUST be available	225: Remote member is already configured, no backups available

Note:

1. Authenticated user who has been authenticated by the web server. This service assumes that a mechanism will be used to bind the authenticated user to management server member.

2. AUTH_TYPE contains the name of authentication method used for authenticating the request. The web server MUST provide this information.

Data Processing:

The data processing specified in this section references member, managed object and account backups tables as specified in the abstract data model.

The management server MUST implement the following rules for generating response content:

1. The response MUST contain account backup data, if the member is active and an account backup is available.

2. The response MUST contain identity template and policy managed objects associated with member, if the member is in a pending state.

Results:

If no faults occur during the operation, the management server MUST return **AutoAccountConfigurationResponse** message as defined in section 2.2.3.10.

3.2.5.2.1.3 AutoActivation

This method provides a mechanism for the management server to bind a client **Identity** with a management server **Member** based on a web server based authentication mechanism. This method MUST be received on an authenticated port.

Request Validation:

Parameter / Web server	Validation conditions	Fault code
DomainGUID	The domain GUID MUST belong to a domain on the management server	209: Domain not found
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as key	200: Account not found
Authenticated user	Authenticated member MUST be identifiable in management server.	401: Activation code not found
Authenticated user	Authenticated member MUST be identifiable in management server and belong to the domain identified by Domain GUID.	401: Activation code invalid
Authenticated user	The authenticated member status MUST be in a pending state	402: Activation code already enrolled

Data Processing:

The data processing specified in this section references member and managed objects tables specified in the abstract data model.

The response MUST contain the configuration code of the authenticated member.

Results:

If no faults occur during the operation, the management server MUST return an **AutoActivationResponse** message as defined in section 2.2.3.12.

3.2.5.2.2 CreateAccount

This method provides a mechanism for the management server to register a client account.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	The DomainGUID MUST belong to a domain on the management server	209: Domain not found
AccountGUID	There MUST be an entry in the Accounts table with AccountGUID and DomainGUID as key	200: Account not found
Payload	There MUST be no security errors in processing the payload	205: Unknown security error processing event
CSMKey	This parameter MUST be present and not empty	204: A required parameter was missing or invalid

Data Processing:

The data processing specified in this section references the domains and accounts tables specified in the abstract data model.

The management server MUST locate the domain using domain GUID as the key; the management server MUST obtain the secret key by decrypting CSMKey using the domain's private key.

The management server MUST create a new entry in the accounts table with domain GUID, account GUID, and secret key.

If the account is a device account, the management server MUST create a new device account table entry.

Results:

If no faults occur during the operation, the management server MUST return a **CreateAccountResponse** message as defined in section 2.2.3.20.

The following sections specify the processing sequences and rules for **CreateAccount** method.

3.2.5.2.2.1 Request Parsing

The server MUST follow the **CreateAccount** request definition in section 2.2.3.19 to parse the envelope. This includes base64-decoding the **Payload** element.

If the parsing fails, or any required element or attribute is missing, a SOAP fault response MUST be returned. The fault code is as defined in section 2.2.2.2.15, with the value set to either "105", "203" or "204". The fault string MAY be any string describing the failure.

The server MUST save the **DomainGUID** attribute. This identifies the domain the account is to be created in.

3.2.5.2.2.2 Parse Security Elements

The public key information element "urn:groove.net:Cert" MUST be a child of the security element "urn:groove.net:SE" as defined in section 2.2.3.19.1 for **CreateAccountSEType** and MUST have the attributes as defined in section 2.2.3.19.1 for **CertType**.

The following attributes MUST be parsed and saved as the client's security information:

- **EPKAlgo** attribute is the encryption public key algorithm.
- **EPubKey** attribute is the encryption public key, DER encoded as defined in section 3.1.2.
- **EncAlgo** attribute is the encryption algorithm.
- **SPKAlgo** attribute is the signature public key algorithm.
- **SPubKey** attribute is the signature public key, DER encoded as defined in section 3.1.2.
- **SigAlgo** attribute is the signature algorithm.

  The signature algorithm name and the signature key algorithm name MUST both be "RSA". The encryption algorithm and the encryption key algorithm names MUST be either "ELGAMAL" and "DH", or "RSA" and "RSA", respectively.

3.2.5.2.2.3 Decrypt Shared Key

The authenticator element "urn:groove.net:Auth" MUST be a child of the security element "urn:groove.net:SE" as defined in section 2.2.3.19.1 for **CreateAccountSEType** and MUST have the attributes as defined in section 2.2.3.19.1.

The following attributes MUST be parsed and saved as inputs to data integrity verification:

- **Sig** attribute is the message signature

The encrypted key attribute **CSMKey** MUST be an attribute of the security element "urn:groove.net:SE" as defined in section 2.2.3.19.1 for **CreateAccountSEType**. Its value MUST be parsed and saved as inputs to shared key decryption.

The encrypted key MUST be decrypted using the domain's encryption private key with the RSA algorithm, as defined in [PKCS1]. The result is saved as the shared key for the client account and the server domain, indexed with the client account GUID and the domain GUID.

3.2.5.2.2.4 Delete Authenticator Element

The authenticator element "urn:groove.net:Auth" MUST be deleted as the child of the Security element "urn:groove.net:SE", which is part of the fragment element.

Once this is done, the fragment element becomes a header element, as defined in section 3.1.5.1.

3.2.5.2.2.5 Serialize Header Element

The header element MUST be serialized as defined in section 3.1.5.2.

3.2.5.2.2.6 Compute Message Digest

The message digest MUST be computed using the SHA1 algorithm, as defined in [RFC3174] by following these steps:

1. Compute a digest of the serialized message using the SHA1 algorithm.
2. Compute the digest of the digest produced in step 1 using the SHA1 algorithm.

Then the digest produced in step 2 is ready to be signed using RSA as specified in section 3.2.5.2.2.7.

The message digest MUST include a serialized header element (as defined in section 3.2.5.2.2.5).

3.2.5.2.2.7 Verify Signature

The message signature MUST be verified using the client's signature public key, as obtained in section 3.2.5.2.2.2 with the RSA algorithm, as defined in [PKCS1].

The message signature MUST be for the message digest, as computed in section 3.2.5.2.2.6.

Comparison of this message signature with the one saved from section 3.2.5.2.2.2 MUST be performed. If the signatures do not match, the data integrity and authenticity verification has failed. In this case, the message MUST be rejected and the server MUST return a Fault message. The fault code is defined in section 2.2.2.2.15 with value set to "204" or "205", and the fault string can be any string describing the error.

3.2.5.2.2.8 Process Request and Send Response

Upon successfully opening the secured the request, the server processes the request and prepares a response.

The server MUST follow the definition for Create Account response as defined in section 2.2.3.20 to prepare the response envelope. The return code "0" MUST be set.

Because there is no other data to return, the response is not secured.

The server then serializes the envelope and returns to the client.

3.2.5.2.3 DomainEnrollment

This method provides a mechanism for the management server to sign the member's contact using a domain certificate and to set the member's status to active.

Request Validation:

Parameter	Validation conditions	Fault code
KeyID	MUST be valid and present on the server	401: activation code not found
Payload	No security or parsing errors	205: Unknown security error processing event.
SPubKey, SigKeyAlgo, SigAlgo, ActivationKeySignature	Validate the value of the ActivationKeySignature attribute.	403: Signature verification failed during enrollment.

Data Processing:

The data processing specified in this section references domain, account, and member tables specified in the abstract data model.

The management server MUST locate the member using KeyID as key. The management server MUST:

1. Set the member's status to active.

2. Update the member entry with ContactURL and ContactSecurity.

3. Update the member's identity template managed object.

Validate the value of the ActivationKeySignature attribute as follows:

1. Construct a UNICODE string by concatenating "Activation Key: " including the single space at the end with the activation key itself.
2. Convert this UNICODE string into a byte array, by treating string chars as bytes in little-endian format, not including NULL terminator.
3. Verify the signature using identity's RSA signature public key, the padding is specified as defined in [PKCS1].

The response MUST contain the updated the member's identity template managed object. See section 3.2.5.1.4 for details on generating managed object data.

Results:

If no faults occur during the operation, the management server MUST return domain enrollment response as defined in section 2.2.3.22.

3.2.5.2.4 DomainMigrationStatus

This method provides a mechanism for management server to update the migration status of a Member.

The client only calls this method to report on identities that have identity template with migration.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	The DomainGUID MUST belong to a domain on the management server	209: Domain not found
AccountGUID	There MUST be an entry in the Accounts table with AccountGUID and DomainGUID as key	200: Account not found

Data Processing:

The data processing specified in this section references the members table specified in the abstract data model.

The management server MUST set the member status to Migrated if the MigrationStatus is equal to zero and MAY persist **MigratedDomainName** for reporting purposes.

Results:

If no faults occur during the operation, the management server MUST return a **DomainMigrationStatusResponse** message as defined in section 2.2.3.24.

3.2.5.2.5 Enrollment

This method provides a mechanism for the management server to sign a member's contact using a domain certificate and to set the member's status to active.

The client MUST use this method on receipt of server fault code 210.

Request Validation:

Parameter	Validation conditions	Fault code
Domain GUID	There MUST be an entry in the Domains table with Domain GUID as key.	209: Domain not found.
Account GUID	There MUST be an entry in the Accounts table with Account GUID and Domain GUID as key.	200: Account not found.
Domain GUID, Account GUID, Identity URL	There MUST be an entry in the Members table with Domain GUID, Account GUID and IdentityURL as key.	200: Account not found.

Data Processing:

The data processing specified in this section references members and managed objects tables specified in the abstract data model.

The management server MUST locate the member using Domain GUID, Account GUID and IdentityURL as the key. The management server MUST:

1. Set the member's status to active.
2. Update the member entry with ContactURL and ContactSecurity.
3. Update the member's identity template managed object.

The response MUST contain the updated member's identity template managed object. See section 3.2.5.1.4 for details on generating managed object data.

Results:

If no faults occur during the operation, the management server MUST return an **EnrollmentResponse** message as defined in section 2.2.3.26.

3.2.5.2.6 Failures

This method provides a mechanism for the management server to report on device management failures.<7>

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with DomainGUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with AccountGUID and DomainGUID as key.	200: Account not found.

Data Processing:

The failure information MAY be persisted by the management server for reporting purposes. The Failure Code value is 6001. Section 2.2.3.27.1 specifies the failure sub types for the Failure Code 6001.

Results:

If no faults occur during the operation, the management server MUST return a **FailureResponse** message as defined in section 2.2.3.28.

3.2.5.3 Management Services

Client use these sets of management services to:

- Fetch service endpoints from the server.
- Test for identity and policy updates.
- Upload usage statistics.

3.2.5.3.1 GMSConfig

This method provides a mechanism for the management server to ensure clients have the correct protocol and service endpoints.<8>

The protocol server endpoint is formed by appending "/GMSConfig"<9> or "/gms.dll"<10> to the server URL, for example: http://server.example.com/GMSConfig.

The client makes an empty request on server's GMSConfig endpoint to fetch supported protocols and service endpoint. The server in turn updates the response header with supported protocols and service endpoints as follows:

Header Element	Description
ServerVersion	MUST contain a numeric value representing Major version of the server.
NormalProtocol	MUST be "http://" or https://.
NormalPath	MUST contain non-authenticated service endpoint enclosed between "/".
AuthProtocol	MUST be "http://" or https://.
AuthPath	MUST contain authenticated service endpoint enclosed between "/".

3.2.5.3.2 Account Heartbeat

This method provides a mechanism for the management server to identify which accounts are active,

The client sends this message on a regular basis. Regular being defined as at least once in 4 hours. The format for this message is defined at section 2.2.3.1.1.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with Domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with Account GUID and Domain GUID as key.	200: Account not found.
AccountGUID, IdentityURL,	If the request is from a member account (DomainMember is equal to 1), then the status of the member identified by	210: Re-enrollment is

Parameter	Validation conditions	Fault code
DomainMember	Account GUID and Identity URL MUST be active.	required.

Data Processing:

There is no specific data processing requirement. An implementer may choose to mark the identified account as active.

Results:

The management server response MUST return a response of zero for success or the error code formatted as a message defined in section 2.2.3.1.2.

3.2.5.3.3 ManagedObjectStatus

This method provides a mechanism for the management server to ensure clients have the current version of managed objects.

The client sends this message for an identity or for a managed device account.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with Domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with Account GUID and Domain GUID as key.	200: Account not found.
AccountGUID, IdentityURL, DomainMember	If the request is from a member account (DomainMember is equal to 1), then the status of the member identified by Account GUID and Identity URL MUST be active.	210: Re-enrollment is required.

Data Processing:

The data processing specified in this section references domain, accounts, members and managed objects table specified in the abstract data model.

The management server MUST return new or updated managed objects associated with the member or device account. For a member, the managed object response list MUST consist of the identity template managed object and policy managed objects identified by member's IPG GUID. For a device, the managed object response list MUST contain managed objects identified by the device account's DPG GUID.

The management server response MUST contain the following:

- The consistency digest, consistency domain GUID, and consistency identity URL attributes received as part of the request message.
- Managed objects not found in the request or managed objects with newer issued time on the server.
- For a deleted member (status equal to -1), the response managed objects list MUST only contain an entry for the identity object with the **Active** attribute status set to zero.

- For a deleted device (status equal to -1), the response managed objects list MUST only contain an entry for the device object with the **Active** attribute status set to zero.

Results:

If no faults occur during the operation, the management server MUST return a **ManagedObjectStatusResponse** message as defined in section 2.2.3.41.

3.2.5.3.4 ManagedObjectInstall

This method provides a mechanism for management server to record managed object usage information.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Data Processing:

The data processing specified in this section references the members table specified in the abstract data model.

The management server, on receipt of a **ManagedObjectInstall** message for an identity template managed object, MUST

1. Associate an account GUID and identity URL with the member associated with the object GUID.
2. For all other members with the same account GUID and identity URL, set their account GUID and identity GUID attributes to null and their status to pending (1).

Results:

If no faults occur during the operation, the management server MUST return a **ManagedObjectInstallResponse** message as defined in section 2.2.3.39.

3.2.5.3.5 StatisticsPackage

This method provides a mechanism for management server to record client telespace and tool usage information.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Data Processing:

The management server MAY choose to persist statistics information.

Statistics type and payload attribute values:

StatType	Description	String	Long	String2	Long2
0	Tool Time	Tool Template URL	Elapsed Time (Time spent in the tool)	Canonical Telespace URL	Visit Count (number of times user visited the tool)
15	Telespace Time	Canonical Telespace URL	Elapsed Time (Time spent in the space)	Telespace Name	Visit Count (number of times user visited the space)
16	Tool Create	Tool Template URL		Canonical Telespace URL	
17	Telespace Create	Canonical Telespace URL		Telespace Name	
19	Voice over IP	Canonical Telespace URL	Elapsed Time		Use Count (Times the tool was used)
20	Voice over IP - Conference	Canonical Telespace URL	Elapsed Time		Use Count (Times the tool was used)
21	Telespace delete	Canonical Telespace URL			
24	Telespace join	Canonical Telespace URL		Telespace Name	
25	Telespace members	Canonical Telespace URL	Active Members (Number of online / active members in space)	Role	Total Members (Number of members in space)

Results:

If no faults occur during the operation, the management server MUST return a **StatisticsPackageResponse** message as defined in section 2.2.3.47.

3.2.5.4 Backup Services

These services enables client to back up account data to the management server. Account backup frequency policy sets the backup interval frequency. A backup frequency of zero disables client from backing up the account.

3.2.5.4.1 AccountStore

This method provides a mechanism for management server to save account backup data.

The client MAY send an account backup fragmented across multiple account store messages. In such cases, backup GUID binds the fragmented backups together with backup index count providing the correct order for rejoining the fragments.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.
BackupIndexCount	In case of fragmented backup, backup index count MUST be 1 or 1 greater than the previous backup index count.	212: Account backup failed

Data Processing:

The data processing specified in this section references the account backups table specified in the abstract data model.

The management server MUST save account backup data.

Results:

If no faults occur during the operation, the management server MUST return an **AccountStoreResponse** message as defined in section 2.2.3.4.

3.2.5.5 Directory Services

Services in this section enable clients to publish or unpublish their contacts, search for contacts, and fetch a contact. An administrator can set an identity contact publishing policy to control identity publishing and unpublishing services.

3.2.5.5.1 IdentityPublish

This method provides a mechanism for management server to persist contacts published by an identity.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Data Processing:

The data processing specified in this section references the members table specified in the abstract data model.

The management server MUST update the member's contact attribute with the new contact published in this message.

Results:

If no faults occur during the operation, the management server MUST return an **IdentityPublishResponse** message as defined in section 2.2.3.35.

3.2.5.5.2 ContactSearch

This method provides a mechanism for the management server to return a list of contacts that meet the requested search criteria.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Data Processing:

The data processing specified in this section references the members table specified in the abstract data model.

The management server MUST return a list of active members (maximum 50) who MUST:

1. Belong to the domain GUID specified in the message.
2. Be in an active state (status = 2).
3. Have published their contact (contact attribute MUST not be null).
4. Full name or last name, or e-mail or first name or ORG state attribute contains the query string specified in the request. If the query string is empty, select the first 50 members that meet the first three conditions.

Results:

If no faults occur during the operation, the management server MUST return the **ContactSearchResponse** message as defined in section 2.2.3.18.

3.2.5.5.3 ContactFetch

This method provides a mechanism for the management server to return one more member contact data.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Parameter	Validation conditions	Fault code
IdentityGUID	There MUST be an entry in the Members table where member GUID is equal to IdentityGUID, the member's status MUST be active, and the member's contact column MUST NOT be null.	207: Exception during Contact fetch

Data Processing:

The data processing specified in this section references members table specified in the abstract data model.

The management server response MUST contain a list of contacts belonging to the members identified by the identity URL.

Results:

If no faults occur during the operation, the management server MUST return a **ContactFetchResponse** message as defined in section 2.2.3.16.

3.2.5.6 Password Management Services

This is a set of methods to enable client users to reset their client account password. Manual and automatic password resets are the two types of password reset methods. The password reset policy defines which type of password reset method is available to a client. The lifetime of a manual passphrase reset request SHOULD be short and the request data deleted after the lifetime expires.

3.2.5.6.1 PassphraseReset

This method provides a mechanism for the management server to initiate the process for manual password reset.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Data Processing:

The data processing specified in this section references the password reset and accounts tables specified in the abstract data model.

The management server MUST:

- Generate a temporary contact by de-serializing the contact element.
- Generate an encoded hash by base64-encoding the admin public key hash.
- Populate the password request table with digest algorithm, encoded hash, encrypted master key, encrypted secret master key, and temporary contact, and set the request status to pending (1).

Results:

If no faults occur during the operation, the management server MUST return a **PassphraseResetRequestResponse** message as defined in section 2.2.3.43.

3.2.5.6.2 PassphraseResetStatus

This method provides a mechanism for the management server to send an update on a manual passphrase reset request.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Data Processing:

The data processing specified in this section references the password reset table specified in the abstract data model.

The management server MUST set the response status as follows:

- 1 if the request is not found using the account GUID as key.
- 1 if the password reset status is not approved (3).
- 2 if the password reset status is pending (1).
- 0 if the password reset status is approved (2), along with the newly re-encrypted master key and re-encrypted secret master key.

Section 3.2.5.1.6 defines the implementation process for manual password reset.

Results:

If no faults occur during the operation, the management server MUST return a **PassphraseResetStatusResponse** message as defined in section 2.2.3.45.

3.2.5.6.3 AutomaticPasswordReset

This method provides a mechanism for management server to do an automatic password reset and send a message with the new password to the member. The client will use authenticated URL returned by GMSConfig service to make this request, if GMSConfig service does not specify an authenticated URL, then the request will be made over http://server/autoactivate/gms.dll URL.

Request Validation:

Parameter	Validation conditions	Fault code
Authenticated User.	Authenticated member MUST be identifiable in management server.	200: Account not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Parameter	Validation conditions	Fault code
IdentityURL	A unique URL to identify a member.	200: Account not found.
CertificatePublicKeyHash	CertificatePublicKeyHash MUST match the hash of the domain's data recovery certificate.	218: Failure during password reset data.

Data Processing:

The data processing specified in this section references the domain table specified in the abstract data model.

The management server MUST

1. If the authenticated user cannot be matched with a management server member, then management server MUST use combination of AccountGUID and IdentityURL to identify the member.
2. Fetch member’s management domain's data recovery certificate.
3. Generate a new random password (temporary password), with at least 21 bytes of randomness.
4. Use the private key from the data recovery certificate to decrypt the encrypted storage master key info using the RSA algorithm as defined in [PKCS1].
5. Take the first 20 bytes of the result, and save it as a verifier for the following comparison.
6. Take the rest of the result, and save it as a master key itself for the following processing.
7. Compute the SHA1 hash of the account GUID, identity URL, and the master key, in this order:
   - AccountGUID and identity URL are Unicode strings, which, for the purposes of the SHA1 hash, MUST be interpreted as byte arrays, not including a NULL-terminator (2 bytes).
8. Compare the verifier saved in step 5 with the result of the hash.
9. If not equal, the management server MUST send a Fault message with a 218 fault code to the client
10. Otherwise, the management server MUST derive a 256-bit AES secret key, by using the PBKDF2 function with the temporary password, no salt, and iteration count of 1 as defined in [PKCS5] (default pseudo-random function HMAC-SHA1 is used in this case).
11. Next, this secret key MUST be used to encrypt both master keys using AES CTR mode. A new initialization vector (IV) MUST be generated when encrypting each key.
12. Compute the SHA1 digest of the following, in order: re-encrypted master key, re-encrypted master key's IV, re-encrypted secret master key, re-encrypted secret master key's IV.
13. Compute HMAC-SHA1 with the computed SHA1 digest, and the secret key used to re-encrypt master keys.
14. Send an e-mail with the temporary password to the e-mail address kept by the server for the user requesting the reset.
15. The response MUST contain:

1. Re-encrypted master keys ("EncryptedMasterKey" and "EncryptedSecretMasterKey")
2. IVs for both re-encrypted master keys
3. Computed HMAC MUST be set as MAC attribute

Results:

If no faults occur during the operation, the management server MUST return an **AutomaticPasswordResetStatusResponse** message as defined in section 2.2.3.14.

The master key is used for encrypting the account keys and the secret master key is used to establish the handshake communication. See section 5 for a description of the master key and secret master key.

3.2.5.7 Auditing Services

Auditing services enables clients to upload audit logs to the management server. The type and the kind of client events that need to be audited are defined by the management server audit policy. Section 3.1.2 defines the security implementation for this set of services.

3.2.5.7.1 AuditLogUploadQuery

This method provides a mechanism for the management server to inform the client on range ranges of uploaded audit log events.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Data Processing:

The data processing specified in this section references the audit devices table specified in the abstract data model.

The management server response MUST contain the device GUID hash and Last Purge Sequence (**_lrh**), set the log range low (**_lrl**) attribute to -1, and set the completed (**_com**) attribute to true.

Results:

If no faults occur during the operation, the management server MUST return an **AuditLogUploadQueryResponse** message as defined in section 2.2.3.8.

3.2.5.7.2 Audit Log Upload

This method provides a mechanism for management server to save audit log event data and inform client about the stored log event range.

The audit log data can be spread across multiple messages. The backup GUID binds the fragments, and backup index count provides the order for combining the fragments.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.
BackupIndexCount	If the backup index count value is greater than 1, it MUST be one more than the previous index count.	216: Audit log upload failed

Data Processing:

The data processing specified in this section references the audit chunk data storage table specified in the abstract data model.

The management server MUST implement the following:

- Save the message data in the audit chunk data storage table.
- If this is the last fragment of audit log data (backup index count MUST be equal to backup fragments count), the management server MUST:
  - Re-assemble the chunked data from the audit chunk data storage table keyed on GUID to form an XML document.
  - Update the [audit devices].[device GUID].[last purge sequence] column with the audit attrib sequence number (attribute **_q**) from the re-assembled XML document.

The management server response MUST contain the following:

- In case of missing chunks or processing failure, or if this is not the last fragment of audit log data, the completed attribute (**_com**) MUST be set to false.
- If this is the last fragment and there are no processing failures, the completed (**_com**) attribute MUST be set to true, log range high (**_lrh**) attribute MUST be set to the value of the "last purge sequence" attribute from the "audit devices" collection, and the log range low (**_lrl**) attribute MUST be set to -1.

Results:

If no faults occur during the operation, the management server MUST return an **AuditLogUploadResponse** message as defined in section 2.2.3.6.

3.2.5.7.3 File Upload Query

This method provides a mechanism for the management server to inform the client of the status of the file it wants to upload.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.

Parameter	Validation conditions	Fault code
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.

Data Processing:

The data processing specified in this section references the audit file storage table specified in the abstract data model.

The management server MUST set the response

- Send file (**_sf**) attribute to the status from the Audit File Storage table using file hash (**_ha**) attribute as the key.
- Completed (**_com**) attribute to true.

Results:

If no faults occur during the operation, the management server MUST return a **FileUploadQueryResponse** message as defined in section 2.2.3.33.

3.2.5.7.4 FileUpload

This method provides a mechanism for the management server to store files from clients on the management server for purpose of auditing.

The file MAY be spread across multiple messages. The backup GUID binds the fragments and the backup index count provides the order for combining the fragments.

Request Validation:

Parameter	Validation conditions	Fault code
DomainGUID	There MUST be an entry in the Domains table with domain GUID as key.	209: Domain not found.
AccountGUID	There MUST be an entry in the Accounts table with account GUID and domain GUID as the key.	200: Account not found.
BackupIndexCount	If the backup index count value is greater than 1, it MUST be one more than the previous index count.	216: Audit log upload failed

Data Processing:

The data processing specified in this section references the audit chunk data storage table specified in the abstract data model.

The management server MUST implement the following:

- Save the message data in the audit chunk data storage table.
- If this is the last fragment of file data (backup index count = backup fragments count), the management server MUST :
- Generate the file by re-assembling the chunked data keyed from the audit chunk data storage table

- Store the re-assembled file in audit file storage with status set to present.

The management server response MUST contain the following:

- In case of missing chunks or processing failure, or if this is not the last fragment of file data, the completed attribute (**_com**) MUST be set to false.
- If this is the last fragment and there are no processing failures, the completed (**_com**) attribute MUST be set to true, and the file hash (**_ha**) attribute MUST contain the file hash.

Results:

If no faults occur during the operation, the management server MUST return a **FileUploadResponse** message as defined in section 2.2.3.31.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Client Details

3.3.1 Abstract Data Model

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

Management Server: An entry in this collection represents a management server. A server URL uniquely identifies a server. The server entry MUST have the following attributes:

- **Server URL:** MUST contain a string representing the HTTP address of the management server. The address MUST use http://*hostname*/gms.dll syntax, where hostname is name of the management server in a domain name form (such as fabrikam.com).<11>

Management Domains: A collection of entries corresponding to the domains that are available on the management server. A domain GUID MUST uniquely identify each entry. Each entry MUST include the following attributes:

- **Domain GUID:** A unique GUID used as the domain identifier.
- **Server URL:** A URL representing the domain's management server.
- **Domain Certificate:** An X.509.v3 certificate as defined in section 3.1.3, containing two sets of 2048-bit RSA public keys, one for encryption and one for signing.

Accounts: Each entry in this collection corresponds to an application-defined entity associated with users and devices. An account GUID MUST uniquely identify an entry in this collection. The account entry SHOULD have the following attributes:

- **Account GUID**: A unique GUID to identify an account.

- **Account Data**: Serialized account data.
- **Master Key, Secret Master Key**: Keys used for securing account data.
- **Backup Enabled**: A Boolean flag, true if backup is enabled.
- **Last Backup Date**: The date of the last account backed up to the management server.
- **Backup Frequency**: An integer representing the backup interval in milliseconds.

Device Account: Represents the computer that is hosting the client. A device GUID MUST uniquely identify a device account. The device account entry SHOULD have the following attributes:

- **Device GUID**: A GUID uniquely identifies a device account.
- **Identity URL:** Identifies the identity associated with this device account.
- **Status:** Represents status of device management. The status MUST be one of the following values:
  - 0: Not managed
  - 1: Managed

Managed Objects: A managed device account MUST contain the following managed objects:

- Account Services Policy
- Data Recovery Policy
- Passphrase Policy
- Component Update Policy

Managed Object Status Check Date: The date of the last test of the managed object status.

Domain Accounts: An entry in this set corresponds to a domain account combination. Domain GUID in conjunction with Account GUID uniquely identifies a domain account entry. Each entry MUST include the following attributes:

- **Domain GUID:** Identifies the domain for the entry.
- **Account GUID:** Identifies the account for the entry.
- **Secret Key:** 192-bit symmetric key used for integrity-protecting and encrypting messages between the client and the management server.

Identity: An identity represents a persona using the client. An account MUST contain one or more identities. An identity URL MUST uniquely identify an entry in this collection. An identity SHOULD have the following attributes:

- **Identity URL:** A unique identifier MUST identify an Identity.
- **Account GUID:** Represents the account to which identity belongs.
- **Domain GUID:** Represents the domain managing the identity.
- **Managed Objects:** MUST contain the following managed objects:

- Identity Template
- Data Recovery Policy
- Domain Trust Policy
- Identity Policy

- **Managed Object Status Check Date:** The date of the last test of the managed object status.
- **Contact**: Contains the member's contact.
- **Identity Relay Devices**: Each entry represents relay servers assigned to an Identity.
- **Identity URL:** Identifies the identity.
- **Device URL:** Relay server URL**.**
- **Sequence:** A number indicating access order of relay servers.

Identity Presence Devices: Each entry represents presence servers assigned to an Identity.

- **Identity URL:** Identifies the identity.
- **Device URL:** Relay server URL.
- **Sequence:** A number indicating access order of presence servers.

Managed Object: Represents a managed object received from the management server. Each managed object MUST have the following attributes:

- **Object GUID:** A unique GUID to identify the object.
- **Object Name**: Name of the object.
- **Object Type:** Type of managed object, which MUST be one of the following:
  - Identity Template
  - Data Recovery Policy
  - Domain Trust Policy
  - Identity Policy
  - Account Services Policy
  - Data Recovery Policy
  - Passphrase Policy
  - Component Update Policy
- **Object Data:** A string representing the managed object.
- **Issued Time:** Represents the time the managed object was created in milliseconds since midnight of 01/01/1970.

Audit Log Events:

- **Account GUID:** GUID of the account sending this data.
- **Sequence number:** Increasing sequence number, unique for each log event.
- **Log Data:** XML document containing the audit event log entry.

Audit Files:

- **Hash:** A unique string representing the digest of the binary file
- **Hash Algo:** Name of the algorithm used for computing the digest.
- **File Data:** File content.

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

The following methods are triggered by higher-layer events:

Higher-Layer Event	Effect on the protocol
End-user makes a request to bind the user's persona to a management server member by providing configuration code and management server URL.	This event triggers a key activation message to the management server.
End-user updates own VCARD details.	This event triggers a contact publish message to the management server.
End-user searches for a contact from the management server contact directory.	This event triggers a contact search message to the management server.
End-user adds a contact from server search result to contact list.	This event triggers a contact fetch message to the management server.
End-user makes a password-reset request.	This event will trigger one of the following methods depending on the manual or automatic password reset policy setting. ▪ Manual password policy will trigger a manual passphrase reset request, followed by periodic manual password reset status messages to the management server until the password is reset or denied, or the request is cancelled by the user. ▪ Automatic password policy will trigger an automatic password request to the management server.

3.3.5 Message Processing Events and Sequencing Rules

The message processing events are the same as defined in section 3.2.5.

3.3.5.1 Common Security Processing Rules

3.3.5.1.1 Common Security Processing Rules for Messages Secured with Key Derived from Account Configuration Code

This section specifies the common security processing rules for securing a request envelope with a shared key derived from an account configuration code, and then opening the response secured with the same key.

This processing applies to **KeyActivation** and **DomainEnrollment** messages.

3.3.5.1.1.1 Derive Key

The client has already received an account configuration code, either via **AutoActivation** messages or through an out-of-band means (e-mail, for example).

The client MUST derive a 160-bit symmetric key by calculating a SHA1 hash (as defined in [RFC3174]) on the account configuration code (for computing SHA1 hash, the account configuration string is treated as a Unicode string. The SHA1 hash is computed by interpreting the string as bytes, in little-endian order, not including the terminating NULL character). This key is used for securing requests and opening secured responses.

3.3.5.1.1.2 Prepare Request

The client MUST prepare a request payload as defined in section 2.2.3 for a specific message.

3.3.5.1.1.3 Secure Response

The request payload becomes the payload element as defined in section 3.1.5.1.

The client MUST also prepare a fragment element with the name "fragment" in the namespace "urn:groove.net", as defined in section 2.2.3 under the request payload sub-section. The **fragment** element MUST have one child element as the payload wrapper element. The payload wrapper element MUST have one child element named "SE" in the namespace "urn:groove.net". The "urn:groove.net:SE" element MUST contain no content and is referred as the security element. This **fragment** element becomes the header element as defined in section 3.1.5.1.

The SHA1 digest (as defined in [RFC3174]) of the key MUST be added as the value for the attribute **KeyID** on the security element "urn:groove.net:SE".

With the fragment element, the request payload, and the key derived from the account configuration code, the client MUST follow the steps in section 3.1.5 to secure the request and restore the serialized secured **fragment** element.

3.3.5.1.1.4 Package Payload into Request Envelope and Send

The client MUST package the secured fragment into a request envelope as defined in section 2.2.3. The client then serializes the envelope and sends it back to the server. The client then waits for the response.

3.3.5.1.1.5 Receive and Parse Response

Upon receiving a response, the client MUST follow the specific response definition in section 2.2.3 to parse the envelope. This includes base64-decoding the Payload element.

If the parsing fails, or any required element or attribute is missing, the client MUST treat this as a bad response. The processing stops here.

The client MUST extract the fragment element per response definition. This fragment element contains the secured content.

3.3.5.1.1.6 Open Secure Content

The client MUST take the fragment element and the shared key, and follow rules specified in section 3.1.6 to open the secured content to get back the response payload element. The shared key here is the key derived from the mechanism specified in section 3.3.5.1.1.1.

The client MUST now process the application response.

3.3.5.1.2 Common Security Processing Rules for Messages Secured with Shared Account Key

This section specifies the common security processing rules for securing a request envelope with a shared account key between a client and a server. There is a user account key and a device account key.

The following messages are secured with the user account key:

- **AccountHeartbeat**
- **ContactFetch**
- **ContactSearch**
- **DomainMigrationStatus**
- **Enrollment**
- **FileUpload**
- **FileUploadQuery**
- **IdentityPublish**

The following messages are secured with the device account key:

- **AutoAccountCodeConfiguration**
- **AutoActivation**
- **AutomaticPasswordReset**
- **CreateAccount**
- **PassphraseResetRequest**
- **PassphraseResetStatus**

The following messages can be secured with either the user account key or the device account key depending on whether the audit log, managed object, or statistics package is associated with the user account or device account:

- **AuditLogUpload**

- **AuditLogUploadQuery**
- **ManagedObjectInstall**
- **ManagedObjectStatus**
- **StatisticsPackage**

This processing applies to all messages except for the **AutoAccountCodeConfiguration**, **CreateAccount**, **DomainEnrollment**, and **KeyActivation** messages.

3.3.5.1.2.1 Prepare Request and Retrieve Key

The client MUST prepare a request payload as defined in section 2.2.3 for a specific message.

The client MUST retrieve the shared account key that it has established with the server via previous **CreateAccount** messages.

3.3.5.1.2.2 Secure Request

The request payload becomes the payload element as defined in section 3.1.6.1.

The client MUST also prepare a fragment element with the name "fragment" in the namespace "urn:groove.net", as defined in section 2.2.3 under the request payload sub-section. The **fragment** element MUST have one child element named "Event" as the event wrapper element. The event wrapper element MUST have one child element named "SE" in the namespace "urn:groove.net". The "urn:groove.net:SE" element MUST contain no content and is referred to as the security element. This fragment element becomes the header element as defined in section 3.1.6.1.

The client MUST fill in all required attributes for the **Event** element, as defined in section 2.2.2.2.7. In particular, the **GUID** and **DomainGUID** attributes are used to identify the shared key.

With the **fragment** element, the request payload and the shared key, the client MUST follow the steps in section 3.1.5 to secure the request and restore the serialized secured **fragment** element.

3.3.5.1.2.3 Package Payload into Request Envelope and Send

The client MUST package the secured fragment into a request envelope as defined in section 2.2.3. The client then serializes the envelope and returns to the server. The client then waits for the response.

3.3.5.1.2.4 Receive and Parse Response

Upon receiving the response, the client MUST follow the specific response definition in section 2.2.3 to parse the envelope. This includes base64-decoding the **Payload** element.

If the parsing fails, or any required element or attribute is missing, the client MUST treat this as a bad response. The processing stops here.

3.3.5.1.2.5 Open Secure Content

If the response carries any payload, it is secured with the same shared key as used in section 3.3.5.2.1.

The client MUST extract the **fragment** element as defined in the specific response definition.

The client MUST take the **fragment** element and the shared key, and follow rules specified in section 3.1.6 to open the secured content to restore the response payload element.

The client MUST now process the application response.

3.3.5.2 Bootstrap Services

A set of services to bootstrap entities involved in this protocol.

3.3.5.2.1 Account Configuration and Creation Services

Account configuration methods enables client to associate an identity with a management server member. The client deployment infrastructure will dictate the method of account configuration used by the client.

3.3.5.2.1.1 KeyActivation

Synopsis:

This method enables a client to bind an **Identity** with a management server **member,** during a manual account configuration or an account conversion process.

Request Preparation:

Follow the following steps to compute KeyID:

1. Compute the SHA1 digest of the account configuration code. For computing the SHA1 hash, the account configuration string is treated as a Unicode string. The SHA1 hash is computed by interpreting the string as bytes, in little-endian order, not including the terminating NULL character.

2. Compute the SHA1 digest of the digest generated in step 1.

The client MUST send a key activation request message as defined in section 2.2.3.36.

Response Processing:

On successful response from management server, client SHOULD create a new identity entry, management domain, and server entry if they do not exist.

3.3.5.2.1.2 AutoAccountCodeConfiguration

Synopsis:

This method automatically binds an **Identity** with a management server **member.** It is triggered by the presence of a GPO policy on the first client initialization or during an automated domain migration.

Request Preparation:

The client MUST send a request message as defined in section 2.2.3.9. The request MUST be sent to an authenticated management server port.

Response Processing:

On successful response from the management server, depending on the response type, the client SHOULD either restore the account or create a new identity entry, management domain and server entry if they do not exist.

3.3.5.2.1.3 AutoActivation

Synopsis:

This method automatically binds an **Identity** with a management server **member**. It is triggered by the presence of an auto activation registry key on first time client initialization.

Request Preparation:

The client MUST send request message as defined in section 2.2.3.11. The request MUST be sent to an authenticated management server port over HTTP.

Response Processing:

On receipt of a fault code 200 response from management server, the client SHOULD send a **CreateAccount** message. The **AutoActivation** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On a successful response from the management server, the client SHOULD send the key activation message using the activation key value from the response.

3.3.5.2.2 Create Account

Synopsis:

This method provides a mechanism for a client to register an account domain pair with the management server. The payload of the method contains the secret key for securing further messages between this account domain pair and the management server.

Request Preparation:

The following steps MUST be followed for generating the **CSMKey** attribute:

1. Generate a 192-bit secret MARC4 key .
2. Encrypt the key generated in step 1 using the management domain's encryption public key.

The client MUST send the request message as defined in section 2.2.3.19.

Response Processing:

A successful management server response does not require any further operations.

The following sections specify the processing sequences and rules.

3.3.5.2.2.1 Generate Shared Key and Prepare Payload and Security elements

The client MUST generate a new shared key to be shared by its account (can be a device or a user account) with the server domain. This MUST be a 192-bit MARC4 symmetric key, as defined in section 3.1.1.

The client MUST create an element named "fragment" in the namespace "urn:groove.net". The client then MUST create an element named "Event" as the child of the **fragment** element, as defined in

2.2.3.19.1 as **CreateAccountEventType**. The following attributes MUST be added to the **Event** element:

- The **Domain** attribute identifies the domain in which the account is to be created.
- The **GUID** attribute identifies the account, either a device or a user.
- The **Encrypted** attribute MUST be set to "1".
- The **IsDeviceAccount** attribute identifies whether this is a device account or a user account. Set it to "1" for a device account, and "0" for a user account.
- The **Created** attribute is the timestamp of the creation time.

The client then MUST create a security element named "SE" in the namespace "urn:groove.net" as the child of the **Event** element, as defined in section 2.2.3.19.1 as **CreateAccountSEType**. It contains no content. It is referred to as the security element.

3.3.5.2.2.2 Add Certificate Information

The certificate information element **Cert** in the namespace "urn:groove.net" MUST be created as a child of the security element "urn:groove.net:SE". The following attributes of the client certificate MUST be added to the certificate element:

- **EPKAlgo** attribute is the encryption public key algorithm. It MUST be "DH" or "RSA".
- **EPubKey** attribute is the encryption public key, encoded as defined in section 3.1.2.
- **EncAlgo** attribute is the encryption algorithm. It MUST be "ELGAMAL" OR "RSA"
- **SPKAlgo** attribute is the signature public key algorithm. It MUST be "RSA".
- **SPubKey** attribute is the signature public key, DER encoded.
- **SigAlgo** attribute is the signature algorithm. It MUST be "RSA".

If **EPKAlgo** is "DH", then **ENCAlgo** MUST be "ELGAMAL". If **EPKAlgo** is "RSA", then **ENCAlgo** MUST be "RSA".

3.3.5.2.2.3 Encrypt Shared Key and Add to Security element

The newly generated shared key (section 3.3.5.2.2.1) MUST be encrypted using the domain's encryption public key with the RSA algorithm, as defined in [PKCS1]. The domain certificate information has been obtained from the previous **KeyActivation** or **AutoAccountCodeConfiguration** message.

The encrypted key MUST then be added as the value for the attribute **CSMKey** to the security element "urn:groove.net:SE".

3.3.5.2.2.4 Serialize Fragment Element

The **fragment** element becomes the header element as defined in section 3.1.5.1. It MUST be serialized as defined in section 3.1.5.2. The result is a serialized header element.

3.3.5.2.2.5 Compute Message Digest

The message digest MUST be computed using the SHA1 algorithm, as defined in [RFC3174] by following these steps:

1. Compute a digest of the serialized message using the SHA1 algorithm.

2. Compute the digest of the digest produced in step 1 using the SHA1 algorithm.

Then the digest produced in step 2 is ready to be signed using RSA as specified in section 3.3.5.2.2.6.

The message digest MUST include the serialized header element (as defined in section 3.3.5.2.2.4).

3.3.5.2.2.6 Compute Signature and Add Authenticator Element

The message signature MUST be computed using the client's signature private key with the RSA algorithm, as defined in [PKCS1].

The Message signature MUST be for the message digest, as defined in section 3.3.5.2.2.5.

The authenticator element **Auth** in the namespace "urn:groove.net" MUST be created as the child of the security element "urn:groove.net:SE", with the following attribute:

- The **Sig** attribute MUST have the signature computed as described in this section.

3.3.5.2.2.7 Serialize Header Element

The client MUST serialize the header element (the **fragment** element) as defined in section 3.1.5.2 to produce the serialized secured payload element.

3.3.5.2.2.8 Prepare Envelope and Send

The client MUST follow the **CreateAccount** request envelope template (as defined in section 2.2.3.19) to prepare a request envelope, with the serialized secured payload element from (section 3.3.5.2.2.7) as the content for the **Payload** element of the **ServiceRequestType2**, as defined in section 2.2.2.2.3.

The client then serializes the request envelope and sends it back to the server.

3.3.5.2.2.9 Parse Response

Upon receiving the **CreateAccount** response message, the client MUST first test whether it is a **Fault** message.

If it is a fault message, the client MUST take proper action. The client processing stops here.

For non-fault messages, the client MUST follow the **CreateAccount** response message definition (section 2.2.3.20) to parse the message.

If any required element or attribute is missing, the client MUST treat this as a bad response. The client processing stops here.

The client now marks that the account key with the server domain has been established.

3.3.5.2.3 DomainEnrollment

Synopsis

This method provides a mechanism for a client to get the member's contact signed by the management server and complete the account configuration process.

Request Preparation:

Steps for computing **CSecurity**:

The client MUST send a request message as defined in section 2.2.3.21. The **DomainEnrollment** messages are secured with the key derived from an account configuration code. The security processing rules are defined in section 3.3.5.1.1.

Generate the value of the ActivationKeySignature attribute as follows:

1. Construct a UNICODE string by concatenating "Activation Key: " including the single space at the end with the activation key itself.
2. Convert this UNICODE string into a byte array, by treating string chars as bytes in little-endian format, not including NULL terminator.
3. Compute the SHA1 hash of the byte array.
4. Compute the SHA1 hash of the hash produced in step 3.
5. Sign the SHA1 hash produced in step 4 with the signature private key of an identity using the RSA algorithm, as defined in [PKCS1].

Response Processing:

On successful response, the client MUST update the identity template managed object and contact of the member.

3.3.5.2.4 DomainMigrationStatus

Synopsis

This method provides a mechanism for the client to inform the management server of the migration status of the member.

Request Preparation:

The client MUST send request message as defined in section 2.2.3.23.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **DomainMigrationStatus** message SHOULD be re-sent on successful completion of **CreateAccount** message.

A successful management server response does not require any further operations.

3.3.5.2.5 Enrollment

This method provides a mechanism for the client to get a member's contact signed by the management server and complete the account configuration process.

Request Preparation:

Steps for computing **CSecurity**:

The client MUST send a request message as defined in section 2.2.3.25.

Response Processing:

A successful management server response does not require any further operations.

3.3.5.2.6 Failures

Synopsis:

This method provides a mechanism for the client to report on device management failures.

Request Preparation:

The following table defines the failure type and sub types:

Failure Code (Type)	Description	Sub Types (SubType)
6001	Device Management Failure	2:Unable to delete / modify registry entry. 3:Unable to set create registry entry. 5:Unable to retrieve device policies from the server. 6: User declined to manage the device.

The client MUST send the request message as defined in section 2.2.3.27.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **Failures** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

A successful management server response does not require any further client operations.

3.3.5.3 Management Services

The client uses this set of management services to:

- Test for identity and policy updates.
- Upload usage statistics.

3.3.5.3.1 GMS Config

Synopsis:

This method provides a mechanism for client to fetch service URI's from management server.

Request Preparation:

The client MUST send an empty request to http://<server>/GMSConfig server URL**.**

Response Processing:

The client uses URLs contained in the header for posting authenticated and non-authenticated transactions to the server. On receipt of a fault code 200 response from server, the client MUST use http://<server URL>/gms.dll for non-authenticated and http://<server URL>/AutoActivate/gms.dll for transactions requiring authentication.

3.3.5.3.2 Account Heartbeat

Synopsis:

This method provides a mechanism for the client to enable the management server to mark an account as active.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.1.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **AccountHeartbeat** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

A successful management server response does not require any further operations.

3.3.5.3.3 ManagedObjectStatus

Synopsis:

This method provides a mechanism for client to test for managed object updates. The client on a periodic basis MUST call this method.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.40.

Response processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **ManagedObjectStatus** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On receipt of a fault code 210 response from the management server, the client SHOULD send an **Enrollment** message to the management server. The **ManagedObjectStatus** message SHOULD be re-sent on successful completion of the **Enrollment** message.

On a successful management server response, the client SHOULD:

- Generate a list of managed objects from the device account if the message was sent from a device, or from the identity if the message was sent from an identity.
- Delete managed objects with an active attribute value of false from the list of managed objects.
- Delete managed objects not in the response from the list of managed objects.
- Add new managed objects from the response to the list of managed objects.
- Replace managed objects in the list with managed objects from the response which have a later issued time.

- Update the device account or the identity's managed object collection with the updated list of managed objects.

3.3.5.3.4 ManagedObjectInstall

Synopsis:

This method provides a mechanism for the client to acknowledge receipt of a managed object to the management server.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.38.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **ManagedObjectInstall** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

A successful management server response does not require any further client operations.

3.3.5.3.5 StatisticsPackage

This method provides a mechanism for client to send client statistics to the management server.

Request Preparation:

For details on statistic types and contents of **string**, **long**, **long2**, and **string2** attributes, see section 3.2.5.3.5.

The client MUST send the request message as defined in section 2.2.3.46.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **StatisticsPackage** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

A successful management server response does not require any further client operations.

3.3.5.4 Backup Services

The backup services enable the client to back up account data to the management server. The account backup frequency policy sets the backup interval frequency. A backup frequency of zero MUST disable the client from backing up the account.

3.3.5.4.1 AccountStore

This method provides a mechanism for the client to store account data on the management server at the frequency defined in the account backup policy.

The client can send an account backup fragmented across multiple account store messages. In such cases, the backup GUID MUST bind fragmented backups together with the backup index count providing the correct order for rejoining the fragments.

Request Preparation:

The account data MUST be broken down into 256 kilobyte fragments to prevent memory exhaustion on the management server.

The client MUST send the request message as defined in section 2.2.3.3.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **AccountStore** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On receipt of a fault code 212 response from the management server, the client SHOULD repeat the archival process from the first account backup fragment with a new backup GUID.

A successful management server response does not require any further client operations.

3.3.5.5 Directory Services

Services in this section enable clients to publish or unpublish their contacts, search for contacts and fetch a contact. An administrator can set an identity contact publishing policy to control identity publishing and unpublishing.

3.3.5.5.1 IdentityPublish

This method provides a mechanism for the client to publish the contact of an identity to the management server.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.34.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **IdentityPublish** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

A successful management server response does not require any further client operations.

3.3.5.5.2 Contact Search

This method provides a mechanism for the client to search for contacts on the management server.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.17.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **ContactSearch** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On a successful server response, the search results SHOULD be returned to the end-user.

3.3.5.5.3 Contact Fetch

This method provides a mechanism for the client to fetch a contact from the management server.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.15.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **ContactFetch** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

3.3.5.6 Password Management Services

This is a set of methods to enable client users to reset the user client account password. Manual and automatic passwords are the two password reset methods. The password reset policy defines which password reset method is available to a client.

3.3.5.6.1 PassphraseReset

This method provides a mechanism for the client to request a manual password reset from the management server.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.42.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message for the device account. The **PassphraseResetRequest** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On a successful server response, the client SHOULD initiate a **PassphraseResetStatus** message to test the status.

3.3.5.6.2 PassphraseResetStatus

This method provides a mechanism for the client to test the status of a manual passphrase reset request.

This method MUST be sent at a regular application-defined interval to the management server after completion of a successful manual passphrase reset request, until the management server responds with a success or a failure status or the end-user cancels the passphrase reset request.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.44.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **PasswordResetStatus** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On a successful server response, the client SHOULD proceed with the reset.

3.3.5.6.3 AutomaticPasswordReset

This method provides a mechanism for the client to send an automatic password reset request to the management server.

Request Preparation:

The content of the automatic password reset e-mail to the end-user is derived from the request parameters. It MUST follow the format as defined in section 2.2.3.13.1.

The client MUST send the request message as defined in section 2.2.3.13.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message for the device account. The **AutomaticPasswordReset** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On successful server response, the client SHOULD proceed with the reset.

3.3.5.7 Auditing Services

Auditing services enable clients to upload audit logs to the management server. The types and kinds of client events that are audited are defined by the management server audit policy. Section 3.1.2 defines the security implementation for these set of services.

3.3.5.7.1 Audit Log Upload Query

This method provides a mechanism for the client to query the management server for the range of audit log events uploaded.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.7.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **AuditLogUploadQuery** message SHOULD be re-sent on successful completion of create account message.

On successful server response, the client SHOULD send an **AuditLogUpload** message with audit log events starting at a sequence number greater than the last purge sequence number from the response. The client MAY purge audit log events with sequence numbers up to the last purge sequence number from the response.

3.3.5.7.2 Audit Log Upload

This method provides a mechanism for the client to upload audit log event data to the management server.

The audit log data MAY be spread across multiple messages. The backup GUID binds the fragments and the backup index count provides the order for combining the fragments.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.5.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **AuditLogUpload** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On receipt of fault code 216 response from the management server, the client SHOULD repeat the archival process from the first sequence number with a new backup GUID.

On a successful server response, the client MAY purge audit log events with sequence numbers up to the last purge sequence number from the response.

3.3.5.7.3 File Upload Query

This method provides a mechanism for the client to query the management server on the status of a file it is uploading.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.32.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **FileUploadQuery** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On a successful server response, if the value of status attribute is:

- 0: the client SHOULD call the **FileUpload** method for uploading the file keyed by the hash.
- 1: the client MAY delete the audit file entry.

3.3.5.7.4 FileUpload

This method provides a mechanism for the client to store audit files on the management server.

A file MAY be spread across multiple messages. The backup GUID binds the fragments and the backup index count provides the order for combining the fragments.

Request Preparation:

The client MUST send the request message as defined in section 2.2.3.30.

Response Processing:

On receipt of a fault code 200 response from the management server, the client SHOULD send a **CreateAccount** message. The **FileUpload** message SHOULD be re-sent on successful completion of the **CreateAccount** message.

On receipt of a fault code 216 response from the management server, the client SHOULD repeat the archival process from the first file fragment with a new backup GUID.

On a successful server response, the client MAY purge audit file entry.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

This section describes a typical scenario involving client requests and management server responses. Message payloads are sent over the wire using base64 encoding with portions of the message data encrypted. The examples in this section display the message payloads as base64 decoded and decrypted for informational purposes.

These examples provide the messages for the following scenarios:

- Start a client that has already been assigned to a management server in a previous session. The client sends **ManagedObjectStatus** and **AccountHeartbeat** requests to the server.
- Use the client to search and fetch contact information from the server. The client sends **ContactSearch** and **ContactFetch** requests to the server.

4.1 Managed Object Status

4.1.1 Managed Object Status Request

```
<?xml version='1.0'?><?groove.net version='1.0'?><SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"><SOAP-
ENV:Body><ManagedObjectStatus><Payload xsi:type="base64">

<?xml version='1.0'?><?groove.net version='1.0'?><g:fragment xmlns:g="urn:groove.net"><Event
DomainGUID="7ymdzshkpai3fgqwdui5c332cmx3532gqenqe9i"
GUID="e2c3smux2b4uhfucu8a3wztus9bsyaz8bqbtk6s" GrooveVersion="4,2,0,2623"
IdentityURL="grooveIdentity://Device" IsDeviceAccount="1"
UniqueID="7ymdzshkpai3fgqwdui5c332cmx3532gqenqe9igrooveIdentity://Device"
UserDeviceGuid="e2c3smux2b4uhfucu8a3wztus9bsyaz8bqbtk6s" UserDeviceName="FABRIKAM"
_EventID="1388401961" created="1199892192"><g:SE><g:Enc
EC="7SHLcolAaC65wR0EJvIgEU0HhwBhwR9By41BIHYOqFAo8RcQBp31c5Infe0SVF27Pt6JpCnMKGaJ6P5i4lPH0NAxl
RTvfzOBy92HwqOtPEKf218YKimwcNBajT2rnf6p0rG6zKCj3RAKnet1iXsW5WXKn1m3ACRVzumbhf/n6eqzCHaDg4GwyQ
N22dcvV8Tr63GwDfQs2pkSh5m6ay4IXkAaM7MExPtAp9Q+UZpwDW8a4F7vqR3+rr+DoFqyvUyfr3Mo/NZcyoKFLya68Ch
Uyh0XEH412aRgFiu7hhxEJA3mT4DeBSB2BiVBbdhzYs7ducg6voX2GkIELpA4ahNBZcmUrTcPYCoI1iC27H6UJmD1emN8
NonfjqYT3b4xnyMimWogiWYOx34QtEfTsD6yXoBcuO3EE85K1RmSfQ3a6GRmcoyjUrGlr4IjSO6e9QJ61AdyHYtIEvA1w
g+OqpBeKAaT6LSJ306w8KMcM97D11iHihSfbcRixk+V+tS1NwZ8sbTdPpQVJn4Go9rIvDzwJdnmZYa4TWCESspZhGQEa0
Z1grT+6iMD6f8yv9w4vMjoLDK9Po2c88f7X5MGIPSnnyEa81iqBAiba1TR0IgKBS2+EgpeMWVrs+RZLvO+RVekeqqfYTp
ENFQtmAa5pwFsu58SPjYxY+cVCHrioa+kVW5NShKOHdb6MYekCV+rqRm2MKT1gkGUA4S1fmz3Za9iMsZNQH/jAe6Tn/ZC
IU43LjxGdTf7444hB185gxH6J5HBzsdwW5C9GKfawWo4imZ/uCydc4M3vSHjWOh5jDCfZ8VfeJU7v7PkPi1LQG5DiPuPT
H0MHPFCxwk7AhlpiH2maClOOo3JrSDldOcKd0xIaMEjs5OwurnPtI7s7dE8SEiGt/tQSJo3HyZcaaH0Mg7lNiEzZ0Rm29
iqWrO36O7Ym6e36mIGNk1AYXTTaMw/1Xb/oo063CG44lm9gUsU/cpONec+PpPD6U2MQLhgDYJwiNsAXbBvX7J8+xdt3wD
H+07OjMxZ69v6eXUaSKvfX+7Ttk/7nCNwMO5PZRHvV1rpiiXvLYsjOLhKAsg3/3MRBweKvPNtKh1JSc8lAQZiHPl9xN08
r2c97eva" IV="Zgj65LvcVzqy0VPcnxxYY6NEi7hzA9t6"/><g:Auth
MAC="7U+RtCZ391vithPT59h+zyAvJxg="/></g:SE></Event></g:fragment> </Payload><Version
xsi:type="xsd:int">4</Version><LastBroadcastProcessed
xsi:type="xsd:int">0</LastBroadcastProcessed><MessageSequenceNumber
xsi:type="xsd:int">0</MessageSequenceNumber></ManagedObjectStatus></SOAP-ENV:Body></SOAP-
ENV:Envelope>
```

4.1.2 Managed Object Status Response

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"><SOAP-
```

```
ENV:Body><ManagedObjectStatusResponse><ReturnCode
xsi:type="xsd:int">0</ReturnCode></ManagedObjectStatusResponse></SOAP-ENV:Body></SOAP-
ENV:Envelope>
```

4.2 Account Heartbeat

4.2.1 Account Heartbeat Request

```
<?xml version='1.0'?><?groove.net version='1.0'?><SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"><SOAP-
ENV:Body><AccountHeartbeat><Payload xsi:type="base64">

<?xml version='1.0'?><?groove.net version='1.0'?><g:fragment xmlns:g="urn:groove.net"><Event
DeviceGUID="e2c3smux2b4uhfucu8a3wztus9bsyaz8bqbtk6s"
DomainGUID="7ymdzshkpai3fgqwdui5c332cmx3532gqenqe9i"
GUID="a6afv5ms7sxxpkzzpfvwbra83av34at3mz6ytds" GrooveVersion="4,2,0,2623"
IdentityURL="grooveIdentity://w7e552zcd2us7uhc7upitakem5j9ezxk@" IsDeviceAccount="0"
UniqueID="7ymdzshkpai3fgqwdui5c332cmx3532gqenqe9igrooveIdentity://w7e552zcd2us7uhc7upitakem5j
9ezxk@" UserDeviceGuid="e2c3smux2b4uhfucu8a3wztus9bsyaz8bqbtk6s" UserDeviceName="FABRIKAM"
_EventID="286456191" created="1199892192"><g:SE><g:Enc
EC="MHxDeAuQkDuCO4X9wp+48GwTahMYfccKpR3xyiQD4LIfazjzDuch4pfm4vo/VbFxkKPuiX8s5fDT4IPBg21PdDxt3
F0X8+SPFn2U2cUs0++qT7yTgKafdmQ=" IV="22lzOygTIwwD4migCpwtld7FFyNylnYo"/><g:Auth
MAC="dDbsbD29yKUCzGo6gkbanwyZJnI="/></g:SE></Event></g:fragment>

</Payload><Version xsi:type="xsd:int">4</Version><LastBroadcastProcessed
xsi:type="xsd:int">0</LastBroadcastProcessed><MessageSequenceNumber
xsi:type="xsd:int">0</MessageSequenceNumber></AccountHeartbeat></SOAP-ENV:Body></SOAP-
ENV:Envelope>
```

4.2.2 Account Heartbeat Response

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"><SOAP-
ENV:Body><AccountHeartbeatResponse><ReturnCode
xsi:type="xsd:int">0</ReturnCode></AccountHeartbeatResponse></SOAP-ENV:Body></SOAP-
ENV:Envelope>
```

4.3 Contact Search

4.3.1 Contact Search Request

```
<?xml version='1.0'?><?groove.net version='1.0'?><SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"><SOAP-ENV:Body><ContactSearch><Payload
xsi:type="base64">

<?xml version='1.0'?><?groove.net version='1.0'?><g:fragment xmlns:g="urn:groove.net"><Event
DomainGUID="7ymdzshkpai3fgqwdui5c332cmx3532gqenqe9i" EventID="823032317"
GUID="e2c3smux2b4uhfucu8a3wztus9bsyaz8bqbtk6s" GrooveVersion="4,2,0,2623" HighPriority="1"
IdentityURL="" IsDeviceAccount="1" UserDeviceGuid="e2c3smux2b4uhfucu8a3wztus9bsyaz8bqbtk6s"
UserDeviceName="FABRIKAM" _EventID="823032317" created="1199895081"><g:SE><g:Enc
```

```
EC="GdQuALQFLfLpxqehEWdK/ywMMwveXviCMFCi+D4qL9w2ASUmZa9hFmgq4uKQl1SpG9/AfzuiddWrdCZmiSSJQ1Wfg
AFF3oVHsfc=" IV="UIiomtorqkUhrSxZYCoORFrxU2zLZSmI"/><g:Auth
MAC="GhczB5Gl1e0L/uEyiRkYpstYd00="/></g:SE></Event></g:fragment>

</Payload><Version xsi:type="xsd:int">4</Version><LastBroadcastProcessed
xsi:type="xsd:int">0</LastBroadcastProcessed><MessageSequenceNumber
xsi:type="xsd:int">0</MessageSequenceNumber></ContactSearch></SOAP-ENV:Body></SOAP-
ENV:Envelope>
```

4.3.2 Contact Search Response

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"><SOAP-
ENV:Body><ContactSearchResponse><ReturnCode xsi:type="xsd:int">0</ReturnCode><Payload
data="PD94bWwgdm….

<?xml version='1.0'?><?groove.net version='1.0'?><g:fragment
xmlns:g="urn:groove.net"><ReturnPayloadWrapper><g:SE><g:Enc EC="EyQg93X…

<?xml version='1.0'?><?groove.net version='1.0'?><ContactSearchResponse Count="6"
Max="50"><Contact City="" CompanyEMail="asdf@asdf.com" EMail="" FirstName="" FullName="asdf"
IdentityGUID="9557EB24-5779-4635-BA83-4C4AD001B4F1"
IdentityURL="grooveIdentity://xwf3pnwqvp78z2iu6z8sj6zjdr9hpxv6@" LastName=""
State=""/><Contact City="" CompanyEMail="ASDF@ASDF.COM" EMail="" FirstName=""
FullName="asdf2" IdentityGUID="52B3E1EC-3EBA-4AD3-932E-8AC31921A056"
IdentityURL="grooveIdentity://4nzpd7sw2x95xpcjx9fcfy846iu7krke@" LastName=""
State=""/><Contact City="" CompanyEMail="asdf@asdf.com" EMail="" FirstName=""
FullName="test3" IdentityGUID="E6DDAA98-A668-4ABD-AAF1-E8DD4A2F9317"
IdentityURL="grooveIdentity://9p27h3xs3zw5tw3ejwn47h7g3fvpehad@" LastName=""
State=""/><Contact City="" CompanyEMail="asdf@asdf.com" EMail="" FirstName=""
FullName="test4" IdentityGUID="7F13742D-CEEF-41EE-AEF3-76F2EF8DE3AD"
IdentityURL="grooveIdentity://v8z88i675a4rkdrhvxk4chi65t5d8en7@" LastName=""
State=""/><Contact City="" CompanyEMail="tswift@activities.grooveqa" EMail=""
FirstName="Patriot3" FullName="Patriot3" IdentityGUID="1630142E-E827-4E4B-8B95-58350FA53794"
IdentityURL="grooveIdentity://2dcq8r7dkbdhhymrghduxzs2g4faa5us@" LastName=""
State=""/><Contact City="" CompanyEMail="asdf@asdf.com" EMail="" FirstName=""
FullName="WedTest" IdentityGUID="6BDF5D0A-ED82-43F8-893A-31836B243E9A"
IdentityURL="grooveIdentity://w7e552zcd2us7uhc7upitakem5j9ezxk@" LastName=""
State=""/></ContactSearchResponse>

IV="MWUjQe4StdgxboIWmhuBWI0BHgpXbEhI"/><g:Auth
MAC="qoXq5D4Dfab3Sp0K2Nnk+fEsb9A="/></g:SE></ReturnPayloadWrapper></g:fragment>

" xsi:type="binary"/></ContactSearchResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

4.4 Contact Fetch

4.4.1 Contact Fetch Request

```
<?xml version='1.0'?><?groove.net version='1.0'?><SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"><SOAP-ENV:Body><ContactFetch><Payload
xsi:type="base64">PD94bWwgdm…

<?xml version='1.0'?><?groove.net version='1.0'?><g:fragment xmlns:g="urn:groove.net"><Event
DomainGUID="7ymdzshkpai3fgqwdui5c332cmx3532gqenqe9i" EventID="2096442320"
```

```
GUID="e2c3smux2b4uhfucu8a3wztus9bsyaz8bqbtk6s" GrooveVersion="4,2,0,2623" HighPriority="1"
IdentityURL="" IsDeviceAccount="1" UserDeviceGuid="e2c3smux2b4uhfucu8a3wztus9bsyaz8bqbtk6s"
UserDeviceName="PATRIOT3" _EventID="2096442320" created="1199895089"><g:SE><g:Enc
EC="fv6yoOd…

<?xml version='1.0'?><?groove.net version='1.0'?><ContactFetch><IdentityList><IdentityList
IdentityGUID="9557EB24-5779-4635-BA83-4C4AD001B4F1"/><IdentityList IdentityGUID="52B3E1EC-
3EBA-4AD3-932E-8AC31921A056"/><IdentityList IdentityGUID="1630142E-E827-4E4B-8B95-
58350FA53794"/><IdentityList IdentityGUID="E6DDAA98-A668-4ABD-AAF1-
E8DD4A2F9317"/><IdentityList IdentityGUID="7F13742D-CEEF-41EE-AEF3-
76F2EF8DE3AD"/><IdentityList IdentityGUID="6BDF5D0A-ED82-43F8-893A-
31836B243E9A"/></IdentityList></ContactFetch>

IV="FDTtpyo+pAhQN9iEUyQf+kdbCWm3vZVT"/><g:Auth
MAC="9sx8T5AhK4ufmflV9pjF69+4VD4="/></g:SE></Event></g:fragment>

</Payload><Version xsi:type="xsd:int">4</Version><LastBroadcastProcessed
xsi:type="xsd:int">0</LastBroadcastProcessed><MessageSequenceNumber
xsi:type="xsd:int">0</MessageSequenceNumber></ContactFetch></SOAP-ENV:Body></SOAP-
ENV:Envelope>
```

4.4.2 Contact Fetch Response

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"><SOAP-
ENV:Body><ContactFetchResponse><ReturnCode xsi:type="xsd:int">0</ReturnCode><Payload
data="PD94b…

<?xml version='1.0'?><?groove.net version='1.0'?><g:fragment
xmlns:g="urn:groove.net"><ReturnPayloadWrapper><g:SE><g:Enc EC="mVREQ…

<?xml version='1.0'?><?groove.net version='1.0'?>
<IdentityList IdentityCount="6">
<Identity IdentityGUID="9557EB24-5779-4635-BA83-4C4AD001B4F1" VCard="QkVHSU46V…/>
<Identity IdentityGUID="E6DDAA98-A668-4ABD-AAF1-E8DD4A2F9317" VCard="QkVHSU46Vk…/>
<Identity IdentityGUID="7F13742D-CEEF-41EE-AEF3-76F2EF8DE3AD" VCard="QkVHSU46Vk…/>
<Identity IdentityGUID="6BDF5D0A-ED82-43F8-893A-31836B243E9A" VCard="QkVHSU46Vk…/>
<Identity IdentityGUID="52B3E1EC-3EBA-4AD3-932E-8AC31921A056" VCard="QkVHSU46Vk…/>
<Identity IdentityGUID="1630142E-E827-4E4B-8B95-58350FA53794" VCard="QkVHSU46Vk…/>
</IdentityList>

IV="i0FNdmgDqh21P8CCSpGCqF8951pbEHxR"/><g:Auth
MAC="ZsL6HXQWV9xvPN2WkbW4j5Y2ub8="/></g:SE></ReturnPayloadWrapper> </g:fragment>

xsi:type="binary"/></ContactFetchResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

4.5 XML Serialization

The following example shows the result of serializing a secured fragment element (extra white spaces added for display purposes), as defined in section 3.1.5.2.

```
<?xml version='1.0'?><?groove.net version='1.0'?>
<g:fragment xmlns:g="urn:groove.net">
 <Payload ManagementServer="Fabrikam/gms.dll" Method="Registration">
    <g:SE>
```

```
     <g:Enc
EC="CnF8owJ2L/0hdYe+H3MMF175Doq6A7V+jB4salyupaloRVa5/+bu387bk3CQyz5g1fS2GD61oujJbMjpOuGw41ZtH
/tf0BtrQHGJ15CAY9UjIxVgBf2nmi3xg0wqjZFTQxdqgb31ZQKsB+SqrByCu0sojyehedk2yH14ss3IdtQW4GDNAGrzYk
7Ac9E4dCoJ57irfRZicA=="
 IV="tWcSBZ2ThN5sAk2Hs2XKovupsAA="/>

    <g:Auth MAC="IjE/R8ItYgetn75ecUzJ+gHcYxo="/>
     </g:SE>
 </Payload>
</g:fragment>
```

5 Security

5.1 Security Considerations for Implementers

This protocol has the following security limitations:

- Use of semi-weak algorithms. Uses SHA1 and HMAC-SHA1 when computing message digest and keyed message digest.
- Use of weak algorithms. Uses MARC4 (RC4-drop(256)) for symmetric key encryption.
- Use of non-standard/suspect algorithms. The current protocol uses RSA or ElGamal for public key encryption.
- Insufficient encryption of protocol messages. The current protocol does not encrypt the message header. This allows an attacker to read the data in the message header. The current protocol does not encrypt, sign, or protect the integrity of the return code of a response.
- Use of the same key for encryption and MAC. The current protocol uses the same secret key for both encryption and integrity protection, exposing transmissions to related key attacks.
- Lack of nonce or sequence number to prevent replay attacks. The current protocol does not include a nonce or sequence number in each message to prevent replay attacks. This allows an active attacker to replay messages captured in the past.
- Auto Account Code Configuration requires HTTPS for encryption. Auto Account Code Configuration depends on the link layer security.

5.2 Index of Security Parameters

Security Parameter	Section
Client device encryption public key	1.3.2, 3.1.2, 2.2.3.19.1, 3.2.5.2.2.2, 3.3.5.2.2.2
Client device encryption private key	3.3.1
Client device signature public key	1.3.2, 2.2.3.19.1, 3.2.5.2.2.2, 3.3.5.2.2.2
Client device signature private key	1.3.2, 1.4, 3.3.5.2.2.6
Identity and contact encryption public key	2.2.2.2.6, 3.1.2, 3.2.5.1.6
Identity encryption private key	3.3.1
Identity and contact signature public key	2.2.2.2.6
Identity signature private key	3.2.5.2.2.3
Account encryption public key	1.3.2, 2.2.3.19.1, 3.1.2, 3.2.5.2.2.2, 3.3.5.2.2.2
Account encryption private key	3.3.1
Account signature public key	1.3.2, 2.2.3.19.1, 3.2.5.2.2.2, 3.3.5.2.2.2, 3.3.5.2.2.6
Account signature private key	1.3.2 , 1.4

Security Parameter	Section
Management domain encryption public key	1.3.2, 1.4, 2.2.3.42.1, 3.1.3, 3.1.4, 3.2.1, 3.3.1, 3.3.5.2.2.3
Management domain encryption private key	1.3.2, 3.1.4, 3.2.1, 3.2.5.2.2.3
Management domain signature public key	2.2.2.2.10.6, 3.1.4, 3.2.1, 3.2.5.1.4, 3.3.1
Management domain signature private key	3.1.4, 3.2.1, 3.2.5.1.4
Management domain data recovery encryption public key	2.2.3.13.1, 3.1.3, 3.2.1, 3.3.1
Management domain data recovery encryption private key	3.2.1, 3.2.5.1.6, 3.2.5.6.3
Management domain data recovery signature public key	3.2.1, 3.3.1
Management domain data recovery signature private key	3.2.1
Management domain audit encryption public key	3.1.3, 3.1.4, 3.2.1, 3.3.1
Management domain audit encryption private key	3.1.4, 3.2.1
Management domain audit signature public key	3.1.4, 3.2.1, 3.3.1
Management domain audit signature private key	3.1.4, 3.2.1
Secret key shared between the client device and the management domain	1.3.1.11, 1.3.2, 1.4 2.2.3.19.1, 3.1.4, 3.1.5.1, 3.1.5.5, 3.1.6.1, 3.1.6.6, 3.2.1, 3.2.5.1.2, 3.2.5.2.2, 3.2.5.2.2.2, 3.2.5.2.2.3, 3.3.1, 3.3.5.1.2.2, 3.3.5.1.2.5, 3.3.5.2.2, 3.3.5.2.2.1, 3.3.5.2.2.3, 3.3.5.2.2.9
Secret key shared between the account and the management domain	1.3.1.11, 1.3.2, 1.4, 2.2.3.19.1, 3.1.4, 3.1.5.1, 3.1.5.5, 3.1.6.1, 3.1.6.6, 3.2.1, 3.2.5.1.1, 3.2.5.2.2, 3.2.5.2.2.2, 3.2.5.2.2.3, 3.3.1, 3.3.5.1.2.2, 3.3.5.1.2.5, 3.3.5.2.2, 3.3.5.2.2.1, 3.3.5.2.2.3, 3.3.5.2.2.9
Secret key shared between the account and the management domain for the purpose of account configuration	1.3.2, 1.4, 3.1.5.1, 3.1.5.5, 3.1.6.1, 3.1.6.6, 3.2.5.1.2, 3.3.5.1.1.1, 3.3.5.1.1.6, 3.3.5.2.3
Secret key shared between the account and the management domain for the purpose of audit	2.2.3.5.1, 3.1.4, 3.2.1
Secret key used to encrypt individual chunks of the audit log	2.2.3.5.1, 2.2.3.30.1, 3.1.4
Secret key encryption algorithm for	3.1.1, 3.1.4, 3.1.5.5, 3.1.6.6, 3.2.1, 3.3.5.2.2, 3.3.5.2.2.1, 5.1

Security Parameter	Section
non-audit related purposes	
Secret key encryption algorithm used for audit and automatic password reset	3.1.4, 3.2.1, 3.2.5.6.3
Management domain public key encryption algorithm	3.1.3, 3.1.4, 3.2.1, 3.2.5.1.6, 3.2.5.2.2.3, 3.3.5.2.2.3
Account, identity, and contact public key encryption algorithm	2.2.2.2.6, 2.2.3.19.1, 3.1.2, 3.2.5.1.6, 3.2.5.2.2.2, 3.3.5.2.2.2, 5.1
Signature algorithm	2.2.2.2.6, 2.2.3.19.1, 3.1.3, 3.1.4, 3.2.1, 3.2.5.1.4, 3.2.5.2.2.2, 3.2.5.2.2.7, 3.2.5.2.3, 3.3.1, 3.3.5.2.2.2, 3.3.5.2.2.6
Hash algorithm	2.2.2.2.6, 2.2.2.2.10.6, 2.2.3.5.1, 2.2.3.13.1, 2.2.3.42.1, 3.1.3, 3.1.5.4, 3.1.6.7, 3.2.1, 3.2.5.1.4, 3.2.5.1.6, 3.2.3, 3.2.5.2.2.6, 3.2.5.6.3, 3.3.5.1.1.1, 3.3.5.1.1.3, 3.3.5.2.1.1, 3.3.5.2.2.5, 5.1
HMAC algorithm	3.1.4, 3.1.5.6, 3.1.6.8, 3.2.5.6.3, 5.1
Password-based key derivation function	3.2.5.6.3
Initialization vector	2.2.2.2.22, 2.2.2.3, 2.2.2.3.15, 2.2.3.5.1, 2.2.3.14.1, 2.2.3.30.1, 3.1.1, 3.1.5.5, 3.1.5.7, 3.1.6.2, 3.1.6.6, 3.2.5.6.3, 4.1.1, 4.2.1, 4.4.1, 4.4.2, 4.5
Message signature	1.3.2, 2.2.3.19.1, 2.2.3.6.1, 2.2.3.8.1, 2.2.3.19.1, 2.2.3.31.1, 3.1.4, 3.2.5.2.2.2, 3.2.5.2.2.7, 3.2.5.2.3, 3.3.5.2.2.6
Managed object signature	2.2.2.2, 2.2.2.2.10.1, 2.2.2.2.10.2, 2.2.2.2.10.3, 2.2.2.2.10.4, 2.2.2.2.10.5, 2.2.2.2.10.6, 2.2.2.2.10.7, 2.2.2.2.10.8, 2.2.2.2.13, 3.2.5.1.4
Message HMAC	2.2.2.2.22, 2.2.2.3, 2.2.2.3.16, 2.2.3.5.1, 2.2.3.14.1, 2.2.3.30.1, 3.1.4, 3.1.5.6, 3.1.5.8, 3.1.6.3, 3.1.6.8, 3.2.5.6.3, 4.1.1, 4.2.1, 4.4.1, 4.4.2, 4.5

6 Appendix A: Message Schemas

6.1 Request Message Schemas

```
<xs:schema xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="Envelope">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Body">
   <xs:complexType>
   <xs:sequence>
    <xs:choice>
    <xs:element ref="AccountHeartbeat"/>
    <xs:element ref="AccountStore"/>
    <xs:element ref="AuditLogUpload"/>
    <xs:element ref="AuditLogUploadQuery"/>
    <xs:element ref="AutoAccountCodeConfiguration"/>
    <xs:element ref="AutoActivation"/>
    <xs:element ref="AutomaticPasswordReset"/>
    <xs:element ref="ContactFetch"/>
    <xs:element ref="ContactSearch"/>
    <xs:element ref="CreateAccount"/>
    <xs:element ref="DomainEnrollment"/>
    <xs:element ref="DomainMigrationStatus"/>
    <xs:element ref="Enrollment"/>
    <xs:element ref="Failures"/>
    <xs:element ref="FileUpload"/>
    <xs:element ref="FileUploadQuery"/>
    <xs:element ref="IdentityPublish"/>
    <xs:element ref="KeyActivation"/>
    <xs:element ref="ManagedObjectInstall"/>
    <xs:element ref="ManagedObjectStatus"/>
    <xs:element ref="PassphraseResetRequest"/>
    <xs:element ref="PassphraseResetStatus"/>
    <xs:element ref="StatisticsPackage"/>
    </xs:choice>
   </xs:sequence>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
  <xs:attribute ref="SOAP-ENV:encodingStyle" use="required"/>
 </xs:complexType>
 </xs:element>
 <xs:attribute name="encodingStyle" type="xs:string"/>
</xs:schema>
```

The referenced child elements of the **Body** element are specified in the following schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<xs:import namespace="http://www.w3.org/1999/XMLSchema-instance"/>
<xs:element name="AccountHeartbeat" type="ServiceRequestType1"/>
<xs:element name="AccountStore" type="ServiceRequestType1"/>
<xs:element name="AuditLogUpload" type="ServiceRequestType1"/>
<xs:element name="AuditLogUploadQuery" type="ServiceRequestType1"/>
<xs:element name="AutoAccountCodeConfiguration" type="ServiceRequestType3"/>
<xs:element name="AutoActivation" type="ServiceRequestType1"/>
<xs:element name="AutomaticPasswordReset" type="ServiceRequestType1"/>
<xs:element name="ContactFetch" type="ServiceRequestType1"/>
<xs:element name="ContactSearch" type="ServiceRequestType1"/>
<xs:element name="CreateAccount" type="ServiceRequestType2"/>
<xs:element name="DomainEnrollment" type="ServiceRequestType3"/>
<xs:element name="DomainMigrationStatus" type="ServiceRequestType1"/>
<xs:element name="Enrollment" type="ServiceRequestType1"/>
<xs:element name="Failures" type="ServiceRequestType1"/>
<xs:element name="FileUpload" type="ServiceRequestType1"/>
<xs:element name="FileUploadQuery" type="ServiceRequestType1"/>
<xs:element name="IdentityPublish" type="ServiceRequestType1"/>
<xs:element name="KeyActivation" type="ServiceRequestType3"/>
<xs:element name="ManagedObjectInstall" type="ServiceRequestType1"/>
<xs:element name="ManagedObjectStatus" type="ServiceRequestType1"/>
<xs:element name="PassphraseResetRequest" type="ServiceRequestType1"/>
<xs:element name="PassphraseResetStatus" type="ServiceRequestType1"/>
<xs:element name="StatisticsPackage" type="ServiceRequestType1"/>

<xs:complexType name="ServiceRequestType1">
<xs:sequence>
 <xs:element name="Payload" type="ContentPayloadType"/>
 <xs:element name="Version" type="NumericType"/>
 <xs:element name="LastBroadcastProcessed" type="NumericType"/>
 <xs:element name="MessageSequenceNumber" type="NumericType"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ServiceRequestType2">
<xs:sequence>
 <xs:element name="Payload" type="ContentPayloadType"/>
 <xs:element name="Version" type="NumericType"/>
 <xs:element name="LastBroadcastProcessed" type="NumericType"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ServiceRequestType3">
<xs:sequence>
 <xs:element name="Payload" type="AttributePayloadType"/>
 <xs:element name="Version" type="NumericType"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="AttributePayloadType">
<xs:attribute name="data" type="xs:base64Binary" use="required"/>
<xs:attribute
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
     ref="xsi:type"
     use="required"
     fixed="binary"/>
</xs:complexType>

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

```
 <xs:simpleContent>
  <xs:extension base="xs:base64Binary">
  <xs:attribute
      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
      ref="xsi:type"
      use="required"
      fixed="base64"/>
  </xs:extension>
 </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="NumericType">
 <xs:simpleContent>
  <xs:extension base="xsd:int">
  <xs:attribute
      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
      ref="xsi:type"
      use="required"
      fixed="xsd:int"/>
  </xs:extension>
 </xs:simpleContent>
 </xs:complexType>
</xs:schema>
```

The referenced "xsi:type" is specified in the following schema:

```
<xs:schema xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:attribute name="type" type="xs:string"/>
</xs:schema>
```

6.2 Response Message Schemas

```
<xs:schema xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import/>
 <xs:element name="Envelope">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="Body">
   <xs:complexType>
   <xs:sequence>
    <xs:choice>
    <xs:element ref="AccountHeartbeatResponse"/>
    <xs:element ref="AccountStoreResponse"/>
    <xs:element ref="AuditLogUploadResponse"/>
    <xs:element ref="AuditLogUploadQueryResponse"/>
    <xs:element ref="AutoAccountCodeConfigurationResponse"/>
    <xs:element ref="AutoActivationResponse"/>
    <xs:element ref="AutomaticPasswordResetResponse"/>
    <xs:element ref="CreateAccountResponse"/>
```

```
    <xs:element ref="ContactFetchResponse"/>
    <xs:element ref="ContactSearchResponse"/>
    <xs:element ref="DomainEnrollmentResponse"/>
    <xs:element ref="DomainMigrationStatusResponse"/>
    <xs:element ref="EnrollmentResponse"/>
    <xs:element ref="FailuresResponse"/>
    <xs:element ref="FileUploadResponse"/>
    <xs:element ref="FileUploadQueryResponse"/>
    <xs:element ref="IdentityPublishReponse"/>
    <xs:element ref="ManagedObjectInstallResponse"/>
    <xs:element ref="KeyActivationResponse"/>
    <xs:element ref="ManagedObjectStatusResponse"/>
    <xs:element ref="PassphraseResetRequestResponse"/>
    <xs:element ref="PassphraseResetStatusResponse"/>
    <xs:element ref="StatisticsPackageResponse"/>
    </xs:choice>
   </xs:sequence>
   </xs:complexType>
  </xs:element>
  </xs:sequence>
  <xs:attribute ref="SOAP-ENV:encodingStyle" use="required"/>
 </xs:complexType>
 </xs:element>
 <xs:attribute name="encodingStyle" type="xs:string"/>
</xs:schema>
```

The referenced child elements of the **Body** element are specified in the following schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:import namespace="http://www.w3.org/1999/XMLSchema-instance"/>

 <xs:element name="AccountHeartbeatResponse" type="ServiceResponseType1"/>
 <xs:element name="AccountStoreResponse" type="ServiceResponseType1"/>
 <xs:element name="AuditLogUploadResponse" type="ServiceResponseType2"/>
 <xs:element name="AuditLogUploadQueryResponse" type="ServiceResponseType2"/>
 <xs:element name="AutoAccountCodeConfigurationResponse"
             type="ServiceResponseType2"/>
 <xs:element name="AutoActivationResponse" type="ServiceResponseType2"/>
 <xs:element name="AutomaticPasswordResetResponse"
             type="ServiceResponseType2"/>
 <xs:element name="CreateAccountResponse" type="ServiceResponseType1"/>
 <xs:element name="ContactFetchResponse" type="ServiceResponseType2"/>
 <xs:element name="ContactSearchResponse" type="ServiceResponseType2"/>
 <xs:element name="DomainEnrollmentResponse" type="ServiceResponseType2"/>
 <xs:element name="DomainMigrationStatusResponse" type="ServiceResponseType1"/>
 <xs:element name="EnrollmentResponse" type="ServiceResponseType1"/>
 <xs:element name="FailuresResponse" type="ServiceResponseType1"/>
 <xs:element name="FileUploadResponse" type="ServiceResponseType2"/>
 <xs:element name="FileUploadQueryResponse" type="ServiceResponseType2"/>
 <xs:element name="IdentityPublishReponse" type="ServiceResponseType1"/>
 <xs:element name="KeyActivationResponse" type="ServiceResponseType2"/>
 <xs:element name="ManagedObjectInstallResponse" type="ServiceResponseType1"/>
 <xs:element name="ManagedObjectStatusResponse" type="ServiceResponseType3"/>
 <xs:element name="PassphraseResetRequestResponse" type="ServiceResponseType1"/>
 <xs:element name="PassphraseResetStatusResponse" type="ServiceResponseType2"/>
 <xs:element name="StatisticsPackageResponse" type="ServiceResponseType1"/>
```

```
<xs:complexType name="ServiceResponseType1">
<xs:sequence>
 <xs:element name="ReturnCode" type="NumericType"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ServiceResponseType2">
<xs:sequence>
 <xs:element name="ReturnCode" type="NumericType"/>
 <xs:element name="Payload" type="AttributePayloadType"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ServiceResponseType3">
<xs:sequence>
 <xs:element name="ReturnCode" type="NumericType"/>
 <xs:element name="ManagedObjects" minOccurs="0" type="AttributePayloadType"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="AttributePayloadType">
<xs:attribute name="data" type="xs:base64Binary" use="required"/>
<xs:attribute
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
     ref="xsi:type"
     use="required"
     fixed="binary"/>
</xs:complexType>

<xs:complexType name="NumericType">
<xs:simpleContent>
 <xs:extension base="xsd:int">
 <xs:attribute
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
     ref="xsi:type"
     use="required"
     fixed="xsd:int"/>
 </xs:extension>
</xs:simpleContent>
 </xs:complexType>
</xs:schema>
```

The referenced "xsi:type" is specified in the following schema:

```
<xs:schema xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:attribute name="type" type="xs:string"/>
</xs:schema>
```

The following specifies the service fault response message schema:

```
<xs:schema xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
  <xs:element name="Envelope">
   <xs:complexType>
    <xs:sequence>
    <xs:element name="Body">
     <xs:complexType>
     <xs:sequence>
      <xs:element name="Fault" type="ServiceFaultResponseType"/>
     </xs:sequence>
     </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:attribute ref="SOAP-ENV:encodingStyle" use="required"/>
   </xs:complexType>
  </xs:element>

  <xs:complexType name="ServiceFaultResponseType">
  <xs:sequence>
   <xs:element name="faultCode" type="xs:int"/>
   <xs:element name="faultString" type="xs:string"/>
  </xs:sequence>
  </xs:complexType>

  <xs:attribute name="encodingStyle" type="xs:string"/>
 </xs:schema>
```

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® Office 2010 suites
- Microsoft® Office Groove® 2007
- Microsoft® Office Groove® Server 2007
- Microsoft® Groove® Server 2010
- Microsoft® SharePoint® Workspace 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.2.2.10.1: 0x04 is only applicable to Office Groove 2007.

<2> Section 2.2.3.40.1: Office Groove 2007 and SharePoint Workspace 2010 set this attribute to a value in the form nnnnn-nnn-nnnnnnn-nnnnn where n is an integer in the range 0 to 9. Office Groove Server 2007 and Groove Server 2010 ignore this attribute if it is specified.

<3> Section 2.2.3.49: SharePoint Workspace 2010 will use the server URLs returned by GMSConfig, if the server version is 14 or higher. In all other cases it will default to Office Groove Server 2007 URLs ("http//<server>/gms.dll" for normal transaction and "https//<server>/AutoActivate/gms.dll" for authenticated transactions). Office Groove 2007 will use Office Groove Server 2007 server URLs.

<4> Section 3.2.1: SharePoint Workspace 2010 will replace the server URL with the URL returned in GMSConfig response.

<5> Section 3.2.3: These server URLs are required by Office Groove 2007. Server URLs for SharePoint Workspace 2010 are dynamically defined in the response to GMSConfig request.

<6> Section 3.2.5.2.1.2: HTTPS for Auto Account Code Configuration is not enforced while the Groove client implementation uses HTTPS for Auto Account Code Configuration request; the Groove management server implementation does not enforce it. That is, the management server does not test that the request is received on top of a SSL link layer.

<7> Section 3.2.5.2.6: This method is implemented in Office Groove Server 2007 only. This method is not implemented in Groove Server 2010.

<8> Section 3.2.5.3.1: This method is implemented in Groove Server 2010 only. This method is not implemented in Office Groove Server 2007.

<9> Section 3.2.5.3.1: Applicable to Groove Server 2010

<10> Section 3.2.5.3.1: Applicable to Office Groove Server 2007 and Groove Server 2010 (for backward compatibility)

<11> Section 3.3.1: SharePoint Workspace 2010 will replace the server URL with the URL returned in GMSConfig response.

8 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

9 Index

T

V

X