Scalix API Guide

Version 11

Revision 2

Scalix API Guide

Published by Scalix, Inc.
149 Madison Ave. Suite 302
New York, NY 10016
USA

Product Version: 11
Document Revision: 2

Notices

The information contained in this document is subject to change without notice.

Scalix Inc. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantibility and fitness for a particular purpose. Scalix Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

UNIX is a registered trademark of The Open Group.

Linux is a registered trademark of Linus Torvalds.

Java is a registered trademark of Sun Microsystems, Inc.

Microsoft and Active Directory are trademarks or registered trademarks of Microsoft Corporation.

All other company names, product names, service marks, fonts, and logos are trademarks or registered trademarks of their respective companies.

Restricted Rights Legend

Use, duplication, or disclosure is subject to restrictions as set forth in contract subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause 52.227-FAR14.

Notice for Open-Source Software

Contents

Introduction to This Guide

About This Guide

This guide outlines the application programming interface (API) of Scalix so that third-parties can integrate their products with Scalix.

For an introduction to Scalix, see the *Introduction to Scalix* and *Scalix Architecture* chapters of the *Scalix Installation Guide*.

How to Use This Guide

This guide uses the following typographical conventions.

Table 1: Conventions Used

Convention	Explanation
<Angle Brackets>	Values that you need to supply are sometimes shown using angle brackets. For example: http://<server_name>/webmail
Code	This smaller font indicates code to write or run. For example: `./scalix-installer`
Italics	Indicates a document or section, a directory path, a file, or the name of a window. For example: Open the */var/opt/scalix* folder. Or: The *Reply* screen appears.

Related Documents

Scalix manuals include:

- *Scalix Release Notes*
- *Scalix Installation Guide*
- *Scalix Migration Guide*
- *Scalix Setup and Configuration Guide*
- *Scalix Client Deployment Guide*
- *Scalix Administration Guide*
- *Scalix API Guide*

In addition, there are online help systems in:

- Scalix Management Console
- Scalix Web Access
- Microsoft Outlook (when enabled for the Scalix connector)

Getting Help

For help with installation, contact technical support at **support@scalix.com**

For the latest documents, see
http://www.scalix.com/community/downloads/documentation.php

For documents, a knowledge base, and forums, see
http://www.scalix.com/community/resources/

Introduction to Scalix APIs

This chapter introduces the Scalix application programming interface (API) for integrating third-party applications with Scalix.

Contents

This chapter includes the following information:

Introduction

Scalix offers messaging and management services application programming interfaces (APIs) to integrate Scalix with third-party desktop and server applications as well as provision users from outside the system, such as through an in-house management console. So there are two types of APIs:

- Messaging Services APIs
- Management Services APIs

Scalix APIs provide the flexibility and extensibility to:

- Integrate with other collaboration and desktop software
- Integrate with legacy customer relationship management (CRM), enterprise resource planning (ERP), billing systems, and more
- Develop your own user management tools
- Automate user management
- Customize reporting of user and data statistics
- Write your own tools or applications
- Automate mail responses

Messaging Services APIs

Messaging Services are server-based APIs for e-mail and calendaring application integration. The APIs allow companies to integrate Linux messaging with their critical applications, such as content management, mobile solutions, CRM, or ERP software. Calendaring functions and data can be integrated directly into other applications, or the data from other applications can be directly integrated into e-mail and calendaring. These APIs use representational state transfer (REST), where REST is a way for XML code to be read in a Web browser.

The Scalix Messaging Services API grants direct access to mailbox data, message delivery, free/busy information, and search, providing the opportunity to easily integrate through one protocol.

The following functionality is available through this API:

- Mailbox access — Access mailbox items, including messages, folders, public folders, and delegate mailboxes
- Message delivery — Send MIME-formatted messages
- Free/busy — Access a user's schedule to determine when they are available
- Search — Full search across a user's private folders, public folders, and delegate mailboxes

The REST approach on which the API is based differs from the Simple Object Access Protocol (SOAP), on which the management services APIs are based, in a number of ways:

- REST is resource-oriented. Every item in the system (messages, folders, and so on) has a URL associated with it
- Given a URL, you can use the standard HTTP requests such as GET, POST, PUT, and DELETE to fetch, create, modify and delete items, respectively
- Additional parameters are part of the request URL
- Content can be returned in several formats

The API's main entry point is: **http(s)://host:port/api**

Management Services APIs

Management Services APIs enable management, administration, and provisioning to enhance the ability to manage resources from the Scalix Management Console (SAC). These APIs are based on the Simple Object Access Protocol (SOAP), which requires a program to serve the data to the end-user.

Scalix Management Services are SOAP-based APIs that enable the management, administration and provisioning of Scalix users and resources from outside the Scalix system. They provide an HTTP/XML interface between the client (Scalix Management Console) and the server's administrative tasks. Other clients that need programmatic interface access to Scalix administrative functionality can use this interface.

Messaging Services APIs

This chapter outlines the messaging application programming interface (API) for integrating third-party applications with Scalix.

Contents

This chapter includes the following information:

Introduction

The Scalix Messaging Services API grants direct access to mailbox data, message delivery, free/busy information, and search, providing the opportunity to easily integrate through one protocol.

Scalix Messaging Services provide the following functionality through this API:

- Mailbox access — Access mailbox items, including messages, folders, public folders, and delegate mailboxes
- Message delivery — Send MIME-formatted messages
- Free/busy — Access a user's schedule to determine when they are available
- Search — Full search across a user's private folders, public folders, and delegate mailboxes

The representational state transfer (REST) approach on which the API is based differs from the Simple Object Access Protocol (SOAP), on which the management services APIs are based, in a number of ways:

- REST is resource-oriented. Every item in the system (messages, folders, and so on) has a URL associated with it
- Given a URL, you can use the standard HTTP requests such as GET, POST, PUT, and DELETE to fetch, create, modify and delete items, respectively
- Additional parameters are part of the request URL
- Content can be returned in several formats

The API's main entry point is:

```
http(s)://host:port/api
```

Terminology

Some terminology that the API uses includes:

Scalix Direct Reference — A 16-character string that uniquely identifies an item (folder or message) in the message store. The message store assigns a direct reference to every item when it is created, and that string never changes, even if the item contents change or the item moves to a different location (the IMAP UID for messages changes however).

In the Scalix Web services API, the direct reference references individual messages. For newly-created items, the direct reference value is returned in the X-Scalix-Directref HTTP header, alongside the standard Location header.

Stateless, stateful — Whether a function remembers preceding events. A stateless event means no previous record is used. Almost all functions in this guide are stateless. A stateful event means that the state is tracked, for example in a storage field specified in the function for that purpose. A login is an example of a stateful function.

URL Schema

Every resource in the Scalix system has a URL assigned to it so that it can be addressed individually. Some examples are:

- Folders
- Messages
- Message parts (MIME parts)

With a few exceptions, Scalix Messaging Services URLs generally follow this structure:

```
http(s)://host:port/api/Max.Mustermann@Company.com/mailbox/INBOX/
00026728e90b589f?output=xml
```

Table 1: URL Syntax

Syntax	Description
http(s)://host:port/api	API Endpoint This is the service's main entry point.
Max.Muster-mann@Company.com	Principal's E-mail Address This is the e-mail address of the person whose content is accessed. The URL schema has to take into account the fact that the person who authenticates against the API is not necessarily accessing their own data (delegation). Therefore the principal's e-mail address is also part of most URLs.
mailbox	Service This is the service being accessed by the API client. Available services are: • userinfo — Returns user information (this service does not have the second component Max.Mustermann@Company.com) • mailbox — Access to folders and messages • search — Mailbox-wide search • delivery — Message delivery • freebusy — User's free/busy data in iCal format
INBOX/ 00026728e90b589f	Resource Path to the item (can be empty for some services).
?output=xml	Query Options Additional options like output, depending on the service.

The basic services you can access via the API are:

- Mailbox
- Search
- Delivery
- Freebusy

Each is outlined here.

Mailbox

The mailbox service provides access to mailbox items:

- Folder list
- Folder contents (message lists)
- Information about individual messages
- Access to individual parts of a message

Typical use cases are:

- List messages in one of several formats: RSS, Atom, JSON, and so on
- Retrieve calendar data in iCal (WebCal)
- Create and modify mailbox items
- Synchronize with other devices

Addressable Items

The main URL for the mailbox service is:

```
http(s)://host:port/api/<email-address>/mailbox
```

Everything following that, up to the optional request parameters, is called the item path and uniquely identifies the item to operate on.

The following table lists all item types and their associated item path structure.

Table 2: Mailbox Syntax

	Item Type	Description	Returned Content Type
1	Folder List	The item path for a folder list is empty. Calling the mailbox service without any additional path information returns a user's folder list. Requesting the folder list returns a list of folders. Every item in that list contains the full URL for that folder. Example: http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox	Depends on output parameter
2	Folder	The item path for a folder is the concatenation of all its parent folders' names plus its local name, all separated by the "/" character. Requesting the folder content returns a list of messages. Examples: http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox/INBOX http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox/lists/evolution-patches http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox/Public Folders/Sales	Depends on output parameter

Table 2: Mailbox Syntax

	Item Type	Description	Returned Content Type
3	Message	The item path for a message is the path of its parent folder plus the "/" character with its direct-reference value appended. Requesting an individual message returns information about that message, including a list of its MIME body parts. Every item in that list contains the full URL to that individual part, which can be used to download that part only in its native content type. Example: http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox/INBOX/0005bd99a3bc7acb	Depends on output parameter
4	MIME Body Part	The item path for an individual MIME body part is the message's item path plus the part specifier, separated by the "/" character. Requesting this URL returns the body part's data in its native content-type. Example: http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox/INBOX/0005bd99a3bc7acb/1.1	body part's content-type
5	MIME Headers	The item path that returns the entire MIME representation of a message (content type message/rfc822) is the message's item path plus "/rfc822". Example: http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox/INBOX/0005bd99a3bc7acb/rfc822	text/plain
6	Entire Mime Message	The item path that returns the MIME headers of a message as plain text is the message's item path plus "/headers". Example: http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox/INBOX/0005bd99a3bc7acb/headers	message/rfc822

Supported Methods

The following table lists the supported HTTP methods.

Table 3: Supported Methods Syntax

Method	Description	Allowed Item Types in URL	Error Codes
GET	Retrieves a mailbox item.	1, 2, 3, 4, 5, 6	200 - OK 400 - Bad Request 401 - Authentication Failure 404 - Item not found
POST	Creates a new folder or message. 1. To create a new message, the whole MIME message has to be provided in the POST request body (content-type set to message/rfc822). 2. To create a new calendar item iCal, data (content-type “text/calendar”) must be provided in the request body. You can use this method to create appointments that do not contain attachments and where the description is plain text. For more complicated items, a complete MIME message has to be constructed. 3. To create a new contact item, the MAPI properties (content-type “application/scalix-properties”) must be provided in the request body as an XML document. See below for a description of the contacts XML schema. 4. To create a new folder, an XML document must be provided in the POST request body specifying the folder name and (optionally) the folder type. Example for creating a calendar folder under the INBOX: POST /api/Max.Mustermann@Company.com/mailbox/INBOX HTTP/1.1 Content-Type: text/xml ... <folder name="Company Events" type="IPF.Appointment"/>	2	201 - Created 400 - Bad request 401 - Authentication Failure 404 - Item (parent folder) not found
PUT	Modifies an existing message, but updates the MAPI properties (calendar and contacts only). Returns the new IMAP UID in the X-Scalix-Uid HTTP header. Content has to be iCal (text/calendar) for appointments or MAPI properties (application/scalix-properties) for contacts.	3	204 - No Content 400 - Bad Request 401 - Authentication Failure 404 - Item (parent folder) not found

Table 3: Supported Methods Syntax

Method	Description	Allowed Item Types in URL	Error Codes
DELETE	Permanently deletes a message.	3	200 - OK 401 - Authentication Failure 404 - Item (parent folder) not found

Optional Query Parameters for the Mailbox Service

The mailbox service supports a number of optional query parameters, mostly to support filtering, sorting, and paging. Using these parameters, a client application can implement a paged approach to listing messages instead of downloading everything at once.

The following table lists optional query parameters.

Table 4: Optional Mailbox Syntax

Parameter	Description
type	Limits output to items of the given IPM types. Valid values are: • IPM.Note • IPM.Post • IPM.Task • IPM.Appointment • IPM.Contact • IPM.DistList • IPM.Schedule.Meeting.Request • IPM.Schedule.Meeting.Resp.Pos • IPM.Schedule.Meeting.Resp.Neg • IPM.Schedule.Meeting.Resp.Tent • IPM.Schedule.Meeting.Canceled • IPM.DeliveryFailure • IPM.DeliveryReceipt • IPM.ReadReceipt
flags	Comma-separated list of flags that limits output to items that have the given flag(s) set. Currently the only supported value is: unread.
sort	Primary sort order. Currently, the only supported value is date.
sort-direction	Direction of sorting. Allowed values are asc (default) and desc.
start	First item to return from entire result set.
end	Last item to return from entire result set.

Example — List the first 10 unread Inbox messages, sorted by date, and formatted as XML:

```
http://hobbit:8080/api/Max.Mustermann@Company.com/mailbox/
INBOX?output=xml&flags=unread&sort=date&start=1&end=10
```

Search

The search service is the front-end to the Scalix Search and Index Service (SIS) and allows full-text queries across entire mailboxes. For example, it is used by the Scalix Web Access e-mail client.

The main URL for that service is:

```
http(s)://host:port/api/<email-address>/search
```

If a folder path is appended, the search is restricted to that folder only. Otherwise it searches the user's entire mailbox.

The following parameters can follow the main URL.

Table 5: Search Syntax

Parameter	Description
q	The search query. This parameter is mandatory and in its simplest form, is just a keyword for which to search. (It results in a compound expression on the subject, from, to and body of the messages.) It can also be a more complicated expression containing named terms combined by the key-words AND and OR. Example 1 — Simple keyword search in the Inbox: http://hobbit:8080/api/Max.Mustermann@Company.com/search/INBOX?q=linuxworld Example 2 — Search for all messages from Nick: http://hobbit:8080/api/Max.Mustermann@Company.com/search?q=from:nick Example 3 — Search for all Inbox message from Nick that contain Scalix in the subject: http://hobbit:8080/api/Max.Mustermann@Company.com/search/ INBOX?q=from:nick%20AND%20subject:scalix
flags	Flags are boolean operators and can be used to further limit the search results. Allowed flags are: flagged, unflagged, seen, unseen, answered, unanswered
start	First result to return. Can be combined with "end" to only return a page of results.
end	Last result to return. Can be combined with "start" to only return a page of results.

Delivery

The delivery service allows sending MIME-formatted messages. It has the fixed URL:

```
http(s)://host:port/api/<email-address>/delivery
```

and only allows the HTTP POST request. The request body must be the entire MIME message to send.

Returned error codes — 202 - OK, 400 - Bad Request

Freebusy

The freebusy service allows you to:

- Query other people's (and your own) free/busy schedule. It returns data formatted as iCal data (GET)
- Update your own free/busy data (POST)

Example — The URL for the free/busy data for user Max.Mustermann@Company.com is:

```
http://hobbit/api/Max.Mustermann@Company.com/freebusy
```

Access to that URL is always authenticated and the principal e-mail address in the URL is of the person whose free/busy data you want to access. You can request your own data.

Example — Returned data:

```
BEGIN:VCALENDAR
BEGIN:VFREEBUSY
ATTENDEE:MAILTO:Max.Mustermann@Company.com
DTSTART:20060531T070000Z
DTEND:20060801T070000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060531T070000Z/20060601T070000Z
FREEBUSY;FBTYPE=BUSYTENTATIVE:20060606T170000Z/20060606T180000Z
FREEBUSY;FBTYPE=BUSY:20060612T153000Z/20060612T160000Z
FREEBUSY;FBTYPE=BUSY:20060613T150000Z/20060613T153000Z
FREEBUSY;FBTYPE=BUSYTENTATIVE:20060613T170000Z/20060613T180000Z
FREEBUSY;FBTYPE=BUSY:20060613T180000Z/20060613T200000Z
FREEBUSY;FBTYPE=BUSY:20060613T204500Z/20060613T213000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060614T070000Z/20060615T070000Z
FREEBUSY;FBTYPE=BUSY:20060614T150000Z/20060614T160000Z
FREEBUSY;FBTYPE=BUSY:20060615T023000Z/20060615T060000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060615T150000Z/20060615T160000Z
FREEBUSY;FBTYPE=BUSY:20060615T160000Z/20060615T180000Z
FREEBUSY;FBTYPE=BUSY:20060615T201500Z/20060615T210000Z
FREEBUSY;FBTYPE=BUSY:20060616T180000Z/20060616T193000Z
FREEBUSY;FBTYPE=BUSYTENTATIVE:20060620T170000Z/20060620T180000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060621T070000Z/20060622T070000Z
FREEBUSY;FBTYPE=BUSY:20060623T020000Z/20060623T043000Z
FREEBUSY;FBTYPE=BUSY:20060624T213000Z/20060625T033000Z
FREEBUSY;FBTYPE=BUSYTENTATIVE:20060627T170000Z/20060627T180000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060628T070000Z/20060629T070000Z
FREEBUSY;FBTYPE=BUSY:20060630T020000Z/20060630T043000Z
END:VFREEBUSY
```

```
END:VCALENDAR
```

Optional Query Parameters for the Freebusy Service

The following table lists optional query parameters.

Table 6: Optional Freebusy Syntax

Parameter	Description
dtstart	Returns free/busy data starting at that time, given as a UTC timestamp (YYYYMMDDTHHMMSSZ). Only valid together with dtend.
dtend	Returns free/busy data up until that time, given as a UTC timestamp (YYYYMMDDTHHMMSSZ). Only valid together with dtstart.

Updating Freebusy Data

Using HTTP POST, you can update your own free/busy data. The supplied data in the POST body has to be of content-type text/calendar and be valid iCal freebusy (similar to what is returned through GET).

Example:

```
POST /api/Max.Mustermann@Company.com/freebusy
Content-Type: text/calendar
...

BEGIN:VCALENDAR
BEGIN:VFREEBUSY
ORGANIZER:MAILTO:Max.Mustermann@Company.com
DTSTART:20060531T070000Z
DTEND:20060801T070000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060531T070000Z/20060601T070000Z
FREEBUSY;FBTYPE=BUSYTENTATIVE:20060606T170000Z/20060606T180000Z
FREEBUSY;FBTYPE=BUSY:20060612T153000Z/20060612T160000Z
FREEBUSY;FBTYPE=BUSY:20060613T150000Z/20060613T153000Z
FREEBUSY;FBTYPE=BUSYTENTATIVE:20060613T170000Z/20060613T180000Z
FREEBUSY;FBTYPE=BUSY:20060613T180000Z/20060613T200000Z
FREEBUSY;FBTYPE=BUSY:20060613T204500Z/20060613T213000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060614T070000Z/20060615T070000Z
FREEBUSY;FBTYPE=BUSY:20060614T150000Z/20060614T160000Z
FREEBUSY;FBTYPE=BUSY:20060615T023000Z/20060615T060000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060615T150000Z/20060615T160000Z
FREEBUSY;FBTYPE=BUSY:20060615T160000Z/20060615T180000Z
FREEBUSY;FBTYPE=BUSY:20060615T201500Z/20060615T210000Z
FREEBUSY;FBTYPE=BUSY:20060616T180000Z/20060616T193000Z
FREEBUSY;FBTYPE=BUSYTENTATIVE:20060620T170000Z/20060620T180000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060621T070000Z/20060622T070000Z
FREEBUSY;FBTYPE=BUSY:20060623T020000Z/20060623T043000Z
FREEBUSY;FBTYPE=BUSY:20060624T213000Z/20060625T033000Z
FREEBUSY;FBTYPE=BUSYTENTATIVE:20060627T170000Z/20060627T180000Z
FREEBUSY;FBTYPE=BUSYUNAVAILABLE:20060628T070000Z/20060629T070000Z
```

```
FREEBUSY;FBTYPE=BUSY:20060630T020000Z/20060630T043000Z
END:VFREEBUSY
END:VCALENDAR
```

Views

Views are simplified versions of mailbox URLs for more common tasks. They are read-only and only respond to GET requests.

Table 7: Views

Parameter	Description
RSS	Returns a folder's items in RSS format. This is just a shortcut for the mailbox service with output set to "rss".
Atom	Returns a folder's items in Atom format. This is just a shortcut for the mailbox service with output set to "atom".
iCal	Returns a folder's items in iCal format. This allows clients that talk Webcal (iCal over HTTP) read-only access to calendar folders. This is just a shortcut for the mailbox service with output set to "ical".

Supported Output Formats

The output format can be specified by setting the "output" query option and applies to the "mailbox" and "search" services.

Example — To list the inbox items as an RSS feed:

```
http://hobbit/api/Max.Mustermann@Company.com/mailbox/INBOX?output=rss
```

When not specified, the Scalix platform API guesses the format. If a browser is used, the output is HTML, and if the "Accepts" HTTP header is set it is used as well. If neither the user-agent nor the Accepts header can be determined, "xml" is used as the default. An existing output query option always takes precedence.

The following table lists understood output formats.

Table 8: Output Formats

Value	Description	Content-Type
xml	Scalix XML schema (see “XML Schema” on page 20)	text/xml
html	A simple HTML representation, mostly useful for debugging and demo purposes	text/html
ical	Renders calendar items in iCal	text/calendar
vcard	Renders contacts as a VCARD	text/vcard
rss	Output is rendered as an RSS feed	application/rss+xml
atom	Output is rendered as an Atom feed	application/atom+xml
json	JavaScript Object Notation (http://www.json.org/)	application/json

XML Schema

The Scalix XML schema used to return information about mailbox items is as follows. A response following this schema always has the general form of:

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.scalix.com/schema" xmlns:xlink="http://
www.w3.org/1999/xlink">
...
</response>
```

Folders Schema

Calling the “mailbox” service without any additional path information returns all listed folders for a user. If “output” is set to “xml”, the response contains one “f” element per folder.

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.scalix.com/schema" xmlns:xlink="http://
www.w3.org/1999/xlink">
 <folder type="IPF.Appointment" dref="00026728e90b589f" ns="private"
name="Calendar"
 special="Calendar" total="10" unread="0" modified="2006-07-16T12:23:10Z"
 xlink:href="http://stingray.us.scalix.com/api/
max.mustermann@company.com/mailbox/Calendar/"/>
 <folder type="IPF.Contact" dref="000267299d0ae7a5" ns="private"
name="Contacts"
 special="Contacts" total="1" unread="0" modified="2006-07-16T12:23:10Z"
 xlink:href="http://stingray.us.scalix.com/api/
max.mustermann@company.com/mailbox/Contacts/"/>
 <folder dref="00026725638a6c36" ns="private" name="Deleted Items"
 special="DeletedItems" total="0" unread="0" modified="2006-07-
16T12:23:10Z"
```

```
 xlink:href="http://stingray.us.scalix.com/api/
max.mustermann@company.com/mailbox/Deleted Items/"/>
 <folder dref="0002672a212c36b2" ns="private" name="Drafts"
 special="Drafts" total="1" unread="0" modified="2006-07-16T12:23:10Z"
 xlink:href="http://stingray.us.scalix.com/api/
max.mustermann@company.com/mailbox/Drafts/"/>
 <folder dref="00020bf25b61027b" ns="private" name="INBOX" total="1392"
unread="0" modified="2006-07-16T12:23:10Z"
 xlink:href="http://stingray.us.scalix.com/api/
max.mustermann@company.com/mailbox/INBOX/"/>
 <folder type="IPF.Journal" dref="000267339a03b448" ns="private"
name="Journal"
 special="Journal" total="0" unread="0" modified="2006-07-16T12:23:10Z"
 xlink:href="http://stingray.us.scalix.com/api/
max.mustermann@company.com/mailbox/Journal/"/>
 <folder dref="000267344d010183" ns="private" name="Junk E-mail"
total="0" unread="0" modified="2006-07-16T12:23:10Z"
 xlink:href="http://stingray.us.scalix.com/api/
max.mustermann@company.com/mailbox/Junk E-mail/"/>
 .
 .
 .
</response>
```

The items returned contain the following information as attributes.

Table 9: Attributes

Attribute	Description	Mandatory?
xlink:href	This folder's item URL.	✓
type	The type of items this folder contains. Although a folder can contain items of any type, this attribute is a hint about what purpose a folder serves. Allowed values include: • IPF.Appointment (calendar items) • IPF.Contact (contact items) • IPF.Journal (journal items) • IPF.Task (task items) • IPF.StickyNote (sticky note items) Use the value as a hint about the type of items in a folder, for example for choosing a specific icon for it when drawing the folder list.	
dref	The value is used as a hint about the type of items in a folder, for example for choosing a specific icon for it when drawing the folder list.	✓
modified	Timestamp of the last modification (item added, modified or deleted).	
ns	The namespace this folder is in. These include: • private — The user's private namespace • shared — Public folders (also known as bulletin boards or shared folders) • others — Folders in the "Other Users" namespace (subscribed mailboxes from other Scalix users)	

Table 9: Attributes

Attribute	Description	Mandatory?
parent	The direct reference value for this folder's parent folder. Top-level folders do not have this attribute set. This value can be used by the client to create a folder hierarchy.	
name	The folder's short name, not including the name of the parent folder.	✓
special	Marks a folder as one of the special folders. Allowed values include: • Calendar (top-level "Calendar" folder) • Contacts (top-level "Contacts" folder) • DeletedItems (top-level "Deleted Items" folder) • SentItems (top-level "Sent Items" folder) • Drafts (top-level "Drafts" folder) • Journal (top-level "Journal" folder) • Tasks (top-level "Tasks" folder) This attribute is optional and a client can use it to pick specific icons for those folders when displaying the folder list.	
total	Number of items in that folder.	✓
unread	Number of unread items in that folder.	✓

Message Schema

When requesting message listings for a particular folder, a list of “m” nodes are returned, one node per message.

The following tables list possible attributes and elements that can be returned per message.

Table 10: Attributes

Attribute	Description	Mandatory?
xlink:href	This folder’s item URL.	✓
type	Item type. Can be one of the following: • IPM.Note • IPM.Post • IPM.Task • IPM.Appointment • IPM.Contact • IPM.DistList • IPM.Schedule.Meeting.Request • IPM.Schedule.Meeting.Resp.Pos • IPM.Schedule.Meeting.Resp.Neg • IPM.Schedule.Meeting.Resp.Tent • IPM.Schedule.Meeting.Canceled • IPM.DeliveryFailure • IPM.DeliveryReceipt • IPM.ReadReceipt	
dref	This item’s direct-reference value.	✓
parent	The parent’s (folder) direct-reference value.	✓
uid	This item’s UID (same as the IMAP UID).	✓

Table 10: Attributes

Attribute	Description	Mandatory?
flags	The decimal numerical value of the bit-field representing the flags of this message. The individual flag values are: 1: deleted 2: seen 4: flagged 8: replied 16: has attachment(s) 32: junk 64: not junk 128: draft 256: forwarded 512: owner 1024: mdn sent 2048: label 1 4096: label 2 8192: label 3 16384: label 4 32768: label 5 A message's flags value is the sum of the individual flag values that are set on that message. For example, a value of 10 means this message was read and replied to.	✓
seen	The seen flag exposed as a separate attribute. Allowed values: true, false.	✓
sent	Sent date (taken from the Date MIME header) in the format yyyy-mm-ddThh:mm:ssZ	✓
received	Received date in the format yyyy-mm-ddThh:mm:ssZ	✓
msgid	The message's message ID (taken from the Message-ID MIME header).	✓
senstivity	Sensitivity setting. Allowed values: personal, private, confidential.	
priority	Priority (or importance) setting. Allowed values: high, low.	
relevance	Search relevance as a number between 0 and 1. Only rendered in search responses.	

Table 11: Elements

Element	Description	Mandatory?
subject	Message subject.	✓
from	Author (taken from the From MIME header).	
sender	Sender (taken from the Sender MIME header).	
to	Recipients (taken from the To MIME header).	
cc	Carbon Copy recipients (taken from the Cc MIME header).	
bcc	Blind Carbon Copy recipients (taken from the Bcc MIME header).	
preview	A one KB text preview of the message.	

When requesting information about one particular message, information about the message's body parts is returned in a parts node. The parts node contains a part child node for every body part of the message.

The following table lists all attributes returned for a part.

Table 12: Attributes Returned

Attribute	Description	Mandatory?
xlink:href	This part's item URL, created by the message's item URL and appending the part's spec value. GETting this URL returns the part's data.	✓
ct	Content type.	
spec	MIME part specifier (1.1, 1.2, and so on).	
fname	The part's filename, if available.	
disp	The part's content disposition ("inline", "attachment", and so on).	
size	Part size in kilobytes.	

Example — Message list:

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.scalix.com/schema" xmlns:xlink="http://
www.w3.org/1999/xlink">
 <item dref="0006176085f14be5" flags="10"
 msgid="<H000009300106cc4.1124987127@MHS>"
 parent="00055ccb164f6a5d" received="2005-08-25T09:25:30Z"
 seen="true" sent="2005-08-25T09:25:27Z" type="IPM.Note" uid="847"
 xlink:href="http://hobbit:8080/api/neal.rut@scalix.com/mailbox/INBOX/
0006176085f14be5">
 <subject>Community Edition Initial Survey Results</subject>
 <from>Hal Steglit <Hal.Steglit@scalix.com></from>
 <sender>Hal Steglit <Hal.Steglit@scalix.com></sender>
```

```
 <to>"+All Employees" <+All.Employees@scalix.com></to>
 <preview>Hi All: 400 people have responded to our survey so far out of
525 people
 emailed. We will be sending more survey emails and I expect we will
 get close to 500 responses when all is done. Below is a summary of the
initial feedback </preview>
 </item>
 <item dref="0001c946c046109e" flags="18"
 msgid="<7487362.1150152830180.JavaMail.root>"
 parent="00055ccb164f6a5d" received="2006-06-12T15:53:50Z"
 seen="true" sent="2006-06-12T15:53:50Z"
 type="IPM.Schedule.Meeting.Request" uid="1064"
 xlink:href="http://hobbit:8080/api/neal.rut@scalix.com/mailbox/INBOX/
0001c946c046109e">
 <subject>ui review w/ jamie</subject>
 <from>James.Black@scalix.com</from>
 <sender>James.Black@scalix.com</sender>
 <to>Boris.Kornis@scalix.com,
 max.mustermann@company.com,
   "Nick Atkin" <Nick.Atkin@scalix.com>
   Sasha.Sterling@scalix.com,
   Dennis.Shu@scalix.com,
 Nelson.Owyan@scalix.com,
  Jonathan.Georges@scalix.com</to>
 <cc>Hal.Stegerlit@scalix.com,
 "Florian von Kurnatowski" <Florian.von.Kurnatowski@scalix.com></cc>
 </item>
 .
 .
 .
</response>
```

Example — Individual message:

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.scalix.com/schema" xmlns:xlink="http://
www.w3.org/1999/xlink">
 <item dref="0006176085f14be5" flags="10"
 msgid="<H000009300106cc4.1124987127@MHS>"
 parent="00055ccb164f6a5d" received="2005-08-25T09:25:30Z"
 seen="true" sent="2005-08-25T09:25:27Z" type="IPM.Note" uid="847"
 xlink:href="http://hobbit:8080/api/neal.rut@scalix.com/mailbox/INBOX/
0006176085f14be5">
 <subject>Community Edition Initial Survey Results</subject>
 <from>Hal Steglit <Hal.Steglit@scalix.com></from>
 <sender>Hal Steglit <Hal.Steglit@scalix.com></sender>
 <to>"+All Employees" <+All.Employees@scalix.com></to>
 <preview>Hi All: 400 people have responded to our survey so far out of
525 people
 emailed. We will send more survey emails and I expect we will
 get close to 500 responses when all is done. Below is a summary of the
initial feedback </preview>
 <parts>
 <part ct="text/plain" disp="inline" enc="quoted-printable"
```

```
 fname="" size="6740" spec="1"
 xlink:href="http://hobbit:8080/api/neal.rut@scalix.com/mailbox/INBOX/
0006176085f14be5/1"/>
 <part ct="text/html" disp="inline" enc="quoted-printable"
 fname="" size="14907" spec="2"
 xlink:href="http://hobbit:8080/api/neal.rut@scalix.com/mailbox/INBOX/
0006176085f14be5/2"/>
 </parts>
 </item>
</response>
```

Custom HTTP Headers

The Scalix Messaging Services API makes use of some non-standard HTTP headers. They include:

- X-Scalix-Flags
- X-Scalix-UID
- X-Scalix-Directref

X-Scalix-Flags

This header can be used in requests (POST or PUT) to the mailbox service to set (or clear) multiple flags on a message item. Valid flags are:

- DELETED — Marks a message as deleted
- FLAGGED — Marks a message as flagged
- SEEN — Marks a message as seen (read)
- ANSWERED — Marks a message as answered
- JUNK — Marks a message as junk
- NONJUNK — Marks a message as not-junk
- DRAFT — Marks a message as draft
- FORWARDED — Marks a message as forwarded
- LABEL1 ... LABEL5 — Marks a message with a label

The “X-Scalix-Flags” HTTP header can take any combination of above flag values. Each value must be prefixes with either “+” (for setting the flag) or “-” (for clearing the flag). Individual flag values are separated by spaces.

Example — To mark a message as read, set the X-Scalix-Flags header on either POST or PUT as follows:

```
X-Scalix-Flags: +SEEN
```

Example — To mark a message as unread and give it a certain label:

```
X-Scalix-Flags: -SEEN +LABEL2
```

X-Scalix-UID

When appending a new message through POST or modifying an existing message through PUT, this response header contains the new IMAP UID of the message.

X-Scalix-Directref

When appending a new item through POST, this response header contains the direct reference value that was assigned to that item.

Using Scalix Web Services

The Scalix Message Services API uses several Scalix Web services. They are:

- Authentication
- Creating New Items
  - E-mail
  - Contact
  - Appointment
- Modifying Items
  - Flags
  - Contact and Appointments
  - Sending Messages

Each is outlined here.

Authentication

Basic HTTP authentication is used to authenticate against the platform API. A session ID is generated and returned in the “JSESSIONID” cookie to the caller after it has authenticated successfully. The caller can then use that session ID in subsequent calls without going through authentication again. The caller can also ignore the session-ID and authenticate with every call. Or it can send both session ID and authentication credentials. In the latter case, the session ID is used first. If it turns out that the session ID is no longer valid, the credentials are used to authenticate and establish a valid session again.

The extra client-server round trip caused by the authentication challenge can be avoided by sending the credentials in the first request.

Usually, though not required, a call to the “userinfo” service will be used to start with. This does three things:

- Authenticates the user through basic HTTP authentication
- Establishes a session (session ID is returned in the response)
- Returns user information (display name, e-mail address, and so on)

In this example, the credentials are passed in the very first request to avoid the extra client-server round trip caused by the HTTP authentication challenge. If the credentials are not passed, a 401 response (authorization required) is generated.

The JSESSIONID cookie returned can and should be used in subsequent calls if possible, as done in the examples.

Example — Request:

```
GET /api/userinfo HTTP/1.1
Host: pippin:8080
Authorization: Basic bWF4IG11c3Rlcm1hbm46cGFzcw==
```

Example — Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=92F8E49755895D70DED0D36F5BD2F36C; Path=/api
Content-Type: text/xml
Transfer-Encoding: chunked
Date: Mon, 07 Aug 2006 18:03:04 GMT

<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.scalix.com/schema" xmlns:xlink="http://
www.w3.org/1999/xlink">
    <smtpaddress>Max.Mustermann@scalix.com</smtpaddress>
    <displayname>Max Mustermann</displayname>
    <userclass>Limited</userclass>
    <mailbox>/max.mustermann@scalix.com/mailbox</mailbox>
</response>
```

Creating New Items

A new item can be an e-mail, contact, or appointment.

E-mail

The following request appends a new e-mail-type item to the user's "Sent Items" folder. Note the direct reference value and the IMAP UID in the response.

Example — Request:

```
POST /api/max.mustermann@scalix.com/mailbox/Sent%20Items HTTP/1.1
Content-Type: message/rfc822
Authorization: Basic bWF4IG11c3Rlcm1hbm46cGFzcw==
Host: pippin:8080
Content-Length: 328

Message-ID: <44D7B35F.1030106@scalix.com>
Date: Mon, 07 Aug 2006 14:40:47 -0700
From: Max Mustermann <max.mustermann@scalix.com>
MIME-Version: 1.0
To:  jane.doe@acme.com
Subject: Lunch?
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

Hi Jane,
```

```
How about lunch tomorrow?

Cheers,
Max
```

Example – Response:

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=2FABAA7DCC18B55B2F3B4C547B382416; Path=/api
X-Scalix-Uid: 3
X-Scalix-Directref: 0001002ab7843364
Location: http://pippin:8080/api/max.mustermann@scalix.com/mailbox
Sent%20Items/0001002ab7843364
Content-Length: 0
Date: Mon, 07 Aug 2006 23:10:23 GMT
```

Contact

This creates an address book entry.

Example – Request:

```
POST /api/max.mustermann@scalix.com/mailbox/Contacts HTTP/1.1
Content-Type: application/scalix-properties
Cookie: JSESSIONID=92F8E49755895D70DED0D36F5BD2F36C
Host: pippin:8080
Content-Length: 1644

<contact>
<subject>Jane Doe</subject>
<message_class>IPM.Contact</message_class>
<fax1_address_type>FAX</fax1_address_type>
<company_phone_number>(650) 123-3333</company_phone_number>
<ttytdd_phone>(650) 123-8888</ttytdd_phone>
<primary_fax></primary_fax>
<home_phone2>(650) 123-6666</home_phone2>
<other_phone_number>(650) 123-8888</other_phone_number>
<car_phone_number>(650) 123-1111</car_phone_number>
<radio_phone_number>(650) 123-2222</radio_phone_number>
<work2_phone_number>(650) 123-4568</work2_phone_number>
<primary_phone_number>(650) 123-1111</primary_phone_number>
<callback_phone_number>(650) 123-2222</callback_phone_number>
<pager_phone_number>(650) 123-2222</pager_phone_number>
<preferred_by_name>Whoever</preferred_by_name>
<email1_display_name>Jane.Doe@acme.com</email1_display_name>
<email1_address>Jane.Doe@acme.com</email1_address>
<email1_address_type>SMTP</email1_address_type>
<email1_address_with_comment>Jane Doe (Jane.Doe@acme.com)<
email1_address_with_comment>
<display_name>Jane Doe</display_name>
<web_page_address>http://www.acme.com</web_page_address>
<selected_mailing_address>2</selected_mailing_address>
<country>United States of America</country>
<zip>94404</zip>
```

```
<state>CA</state>
<mobile_phone_number>(650) 716-2534</mobile_phone_number>
<home_phone_number>(650) 765-4321</home_phone_number>
<work_phone_number>(650) 123-4567</work_phone_number>
<initials>J.D.</initials>
<file_as>Doe, Jane</file_as>
<company_name>Acme</company_name>
<first_name>Jane</first_name>
<middle_name></middle_name>
<last_name>Doe</last_name>
<display_name_prefix>Ms.</display_name_prefix>
<nickname>Jane</nickname>
</contact>
```

Example – Response:

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
X-Scalix-Uid: 3
X-Scalix-Directref: 0001002c8f117214
Location: http://pippin:8080/api/max.mustermann@scalix.com/mailbox/
Contacts/0001002c8f117214
Content-Length: 0
Date: Tue, 08 Aug 2006 00:06:49 GMT
```

Appointment

This creates a calendar entry.

Example – Request:

```
POST /api/max.mustermann@scalix.com/mailbox/Calendar HTTP/1.1
Content-Type: text/calendar
Cookie: JSESSIONID=92F8E49755895D70DED0D36F5BD2F36C
Host: pippin:8080
Content-Length: 802

BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:-//Scalix Inc.//Scalix Server 11.0.0.142-alpha//EN
VERSION:2.0
METHOD:PUBLISH
BEGIN:VEVENT
UID:
040000008200E00074C5B7101A82E00800000000E050E42AA95DC60100000000000000000001
00000008BAD512EBD81924B81D3A97C1C1CE175
LAST-MODIFIED:20060414T080608Z
DTSTAMP:20060808T190000Z
DTSTART:20060808T190000Z
DTEND:20060808T203000Z
TRANSP:OPAQUE
X-MICROSOFT-CDO-BUSYSTATUS:BUSY
SEQUENCE:0
SUMMARY:Lunch with Jane
X-SCALIX-LABEL:0
```

```
LOCATION:Somewhere nice
DESCRIPTION:Meet for lunch
CLASS:PUBLIC
ORGANIZER;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-
ACTION;RSVP=TRUE;CN=Max Mustermann:MAILTO:max.mustermann@company.com
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-
ACTION;RSVP=TRUE;CN=Jane Doe:MAILTO:Jane.Doe@acme.com
END:VEVENT
END:VCALENDAR
```

Example — Response:

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
X-Scalix-Uid: 4
X-Scalix-Directref: 0001002b3d5a4372
Location: http://pippin:8080/api/max.mustermann@scalix.com/mailbox/
Calendar/0001002b3d5a4372
Content-Length: 0
Date: Mon, 07 Aug 2006 23:59:12 GMT
```

Modifying Items

Items that can be modified are flags, contacts, appointments, and messages.

Flags

Flags can be changed (set or unset) with each POST or PUT call on a particular item (except folder). To only change flags on an item, a PUT can be used without any data in the request body.

The following example shows how to change the read status of the above calendar item to “read” (in fact another option was to send the X-Scalix-Flags header alongside the POST request when we created the item).

Example — Request:

```
PUT /api/max.mustermann@scalix.com/mailbox/Calendar/0001002b3d5a4372 HTTP/1.1
Cookie: JSESSIONID=92F8E49755895D70DED0D36F5BD2F36C
Host: pippin:8080
X-Scalix-Flags: +SEEN
```

Example — Response:

```
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Mon, 04 Aug 2008 00:12:12 GMT
```

Contact and Appointments

Modifying an existing contact or appointment is similar to creating one except:

- The method used is PUT
- The proper URL to use is the URL of the item, not the parent folder
- The response only contains the updated IMAP UID (the direct reference value stays the same)

Sending Messages

Sending a message is done by POSTing the entire MIME message (content-type "message/rfc822") to the URL of the delivery service.

Example — Request:

```
POST /api/max.mustermann@scalix.com/delivery HTTP/1.1
Content-Type: message/rfc822
Cookie: JSESSIONID=92F8E49755895D70DED0D36F5BD2F36C
Host: pippin:8080
Content-Length: 328
Message-ID: <44D7B35F.1030106@scalix.com>
Date: Mon, 07 Aug 2006 14:40:47 -0700
From: Max Mustermann <max.mustermann@scalix.com>
MIME-Version: 1.0
To:  jane.doe@acme.com
Subject: Lunch?
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

Hi Jane,

How about lunch tomorrow?

Cheers,
Max
```

Example — Response:

```
HTTP/1.1 202 Accepted
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Mon, 07 Aug 2006 23:04:27 GMT
```

Error Handling

A platform request either returns a valid response (HTTP result code 200 and data) or one of the HTTP error codes, depending on the situation. When an error code is returned, the body of the response contains one line of text, starting with an error code, followed by a whitespace and a fault string.

Example:

```
A00004 wrong username or password
```

Table 13: Error Codes

Error Code	Error Message
A00000	Unknown error
A00001	Unable to resolve server hostname
A00002	Unable to connect to server
A00003	Too many referrals
A00004	Wrong username or password
A00005	Reached maximum number of retries
A00006	Error issuing X-SCALIX-ID command
A00007	Incompatible server version
A00008	Error parsing HTTP authentication header
C00001	Reached maximum number of retries
C00002	Could not set freebusy data
D00001	Could not send message
D00002	Error processing provided MIME message
I00001	Could not initialize IMAP provider
I00002	Could not append message to folder
I00003	Could not create folder
I00004	Could not delete message
I00005	Could not select folder
I00006	Could not fetch message details
I00007	Could not store provided properties
I00008	Could not read content from input stream
I00009	Could not store message flags
M00001	Unable to retrieve folder list

Table 13: Error Codes

Error Code	Error Message
M00002	Unable to retrieve folder list
M00003	Could not set out-of-office settings
M00004	Could not get out-of-office settings
M00005	Full folder sync failed
M00006	Incremental folder sync failed
M00007	Could not get message by its direct reference
M00008	An error occurred when checking the user's database schema
M00009	Search returned an error
M00010	Resource not found
M00011	Error parsing principal's email address
M00012	Folder already exists
M00013	Error while parsing the provided XML data
M00014	Error while parsing the provided MIME data
M00015	Unsupported content-type
M00016	Missing folder information
M00017	Could not get MIME message
M00018	Content-type is not 'text/calendar'
M00019	Error parsing content
M00020	Could not create mailbox item from supplied data
P00001	Could not find renderer
R00001	Invalid request
S00001	Missing parameter 'q'

Management Services APIs

This chapter outlines the application programming interface (API) for integrating third-party applications with Scalix.

Contents

This chapter includes the following information:

Introduction

Management Services are Simple Object Access Protocol (SOAP)-based APIs that enable the management, administration, and provisioning of Scalix users and resources from outside the Scalix system. They provide an HTTP/XML interface between the client (the Scalix Management Console) and the server's administrative tasks. Other clients that need programmatic interface access to Scalix administrative functionality can use this interface.

The SOAP message adheres to the CAA SOAP message specification, except there are no attachments or payload. This is because Scalix APIs incorporate the XML/RPC-like functionality of Scalix Management Console as part of the SOAP envelope instead of encoding into the payload. This simplifies parsing and generating of SOAP messages.

Client Interface

The client HTTP(S) posts SOAP messages to a defined URL for the execution of functions specified as follows. A listener SOAP servlet processes these requests and dispatches them to the appropriate handling modules for processing and final execution on the Scalix server. The way in which these messages eventually get to the Management Agent on the Scalix server is transparent to the client. The Management Services are responsible for handling this delegation.

The credentials for the logged-in administrator identify who is requesting the service. For now, the Management Services are responsible for obtaining the TGT and the Service Ticket for the Management Agent on behalf of the clients. The optional ID is used to identify an instance of the client; this is a unique client-generated ID string.

A general SOAP request message format looks like:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
    <SOAP-ENV:Body>
        <scalix-caa:CAARequestMessage xmlns:scalix-caa="http://
www.scalix.com/caa">
            <ServiceType>scalix.res</ServiceType>
            <Credentials id="client_generated_unique_id">
                <Identity name="xyz" passwd ="U*76%"/>
            </Credentials>
            <FunctionName>FunctionName</FunctionName>
            <ScalixServers>
                <Host>mail.scalix.local</Host>
                <Host>hostname.scalix.local</Host>
            </ScalixServers>
            <FunctionNameParameters>
                <send parameters defined for each function>
            </FunctionNameParameters>
        </CAARequestMessage>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

And a reply looks like this:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
    <SOAP-ENV:Body>
        <scalix-caa:CAAResponseMessage xmlns:scalix-caa="http://
www.scalix.com/caa">
            <ServiceType>scalix.res</ServiceType>
            <FunctionName>FunctionName>FunctionName>
            <ScalixServers>
                <Host>mail.scalix.local</Host>
                ..
                <Host>hostname.scalix.local</Host>
            </ScalixServers>
            <ReturnFunctionNameValues>
                <return set of values defined for each function>
            </ReturnFunctionNameValues>
        </scalix-caa:CAARequestMessage>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A general fault returned by the CAA or Management Services is compliant with SOAP 1.1 and looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>
            <faultstring>CAA Service Error</faultstring>
            <detail>
                <scalix-caa:fault-details xmlns:scalix-caa="http://
www.scalix.com/caa">
                    <message>omaddu : [OM 8265] Authentication ID authid3
already used. </message>
                     <errorcode>OM 8265</errorcode>
                </scalix-caa:fault-details>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

where the <errorcode> OM 8265 </errorcode> is the error returned from the execution of the omadd command.

Client Functions

Each function named as follows maps to a server command to be executed on behalf of the client on the Scalix server by the server's Management Agent. The Management Services offer this functionality as an XML/RPC like Web service and manages this execution via an HTTP/XML protocol. This communication is transparent to the client.

The functions named as follows are categorized into administrative categories. Each function must send its parameters, defined in the sub-schema here, and expect the return values.

Note that except for Login, all functions or methods defined as follows are stateless. That is, no state or data is cached in the Management Services. The LDAP server on the Scalix Server is used for query purposes.

Directory Functions

The methods defined here deal with the Scalix default SYSTEM directory and the hidden USERLIST directory entries.

Users are provisioned in these directories. How information is manipulated is transparent to the client. The Management Services are modified to interact with LDAP or Active Directory via a publicly published and supported administrative interface on these respective servers (which host the directory information).

Except for login, all functions or methods defined in this section are stateless, which means that no record is kept of preceding action so any request must contain all information required.

The basic directory functions you need to access via the API are:

- GetUsersList
- GetUserInfo
- GetExtraUserInfo
- GetUserLoginStatus
- AddUser
- ModifyUser
- DeleteUser
- AddResource
- ModifyResource
- DeleteResource
- GetResourceInfo
- GetResourcesList

Each is outlined here.

GetUsersList

Type — Stateless

Use this function if you want to get Scalix user accounts from remote server(s). The selected user accounts returned depend on the filters provided. Optionally, the return list can be organized by a sort order specified as an element attribute, LAST_NAME, FIRST_NAME or SERVER. The ID is the masterid, which uniquely identifies this user in the Scalix world.

Input parameter subshema:

```
<xsd:element name="GetUsersListParameters"
type="GetUsersListParametersType"/>

<xsd:complexType name="GetUsersListParametersType">
    <xsd:sequence>
        <xsd:element name="filters" type="filtersType"/>
        <xsd:element name="organizedBy"  type="xsd:string" minOccurs="0">
            <xsd:complexType>
                <xsd:attribute name="entity" type="organizedByType”/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="userType" type="userEnumType" use="optional"/>
    <xsd:attribute name="maxRecords" type="xsd:string" use=optional"/>
</xsd:complexType>

<xsd:complexType name="filtersType">
    <xsd:sequence>
        <xsd:element name="cnfilter"  type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="surnamefilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="givennamefilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="intitialsfilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="mailnodefilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="serverfilter" type="xsd:string">
```

```
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="departmentfilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="mailfilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType   name="organizedByType">
    <xsd:sequence>
        <xsd:element name="entity" type="sortEnumType"/>
    <xsd:/sequence>
</xsd:complexType>

<!-- simple type enum definitions -->

<xsd:simpleType name="sortEnumType">
    <restriction base="string">
        <xsd:enumeration value="LAST_NAME"/>
        <xsd:enumeration value="FIRST_NAME"/>
        <xsd:enumeration value="SERVER"/>
    </restriction>
</xsd:simpleType>

<xsd:simpleType name="userEnumType">
    <restriction base="string">
        <xsd:enumeration value="ALL"/>
        <xsd:enumeration value="MAIL"/>
        <xsd:enumeration value="INTERNET"/>
        <xsd:enumeration value="LOGGEDIN"/>
        <xsd:enumeration value="ADMIN"/>
        <xsd:enumeration value="FULL"/>
        <xsd:enumeration value="LIMITED"/>
    </restriction>
</xsd:simpleType>
```

Output parameter subschema:

```
<xsd:element name="ReturnGetUsersListParameters"
type="ReturnGetUsersListParametersType"/>

<xsd:complexType name="ReturnGetUsersListParametersType"/>
    <xsd:sequence>
        <xsd:element name="user" type="xsd:string" minOccurs="0"
maxOccurs="unbounded">
```

```
            <xsd:complexType>
                <xsd:attribute="userAttrType"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="maxRecords" xsd:string" use="optional"/>
</xsd:complexType>

<xsd:complexType name=bucketType>
    <xsd:element name="name" type="xsd:string">
        <xsd:complexType>
            <xsd:attribute name="lastnamefilter" type="xsd:string"
maxOccurs="unbounded"/>
        </xsd:complexType>
    </xsd:element>
</xsd:complexType>

<xsd:simpleType name="userAttrType">
    <xsd:attribute name = "name" type="xsd:string"/>
    <xsd:attribute name ="id" type="xsd:decimal"/>
</xsd:simpleType>
```

Example — Input:

The following filters limit the LDAP search and are used and mapped to their respective LDAP attributes:

- The “lastnamefilter”, “firstnamefilter” and “intialfilter” map to “surname”, “given-Name”, and “initials” LDAP attributes, respectively
- The usertypefilter stipulates whether you want ALL, MAIL, INTERNET, ADMIN, or LOGGE-DIN.
- ALL indicates all users collectively in the enterprise
- MAIL indicates only local Scalix users
- INTERNET indicates only Internet users or contacts without Scalix mailboxes
- ADMIN type is for full Scalix administrators
- LOGGEDIN means users who are currently logged in to the Scalix server

The attribute “maxRecords” is optional. When specified, only maxRecords are returned. If the actual result that matches the search criteria is greater than maxRecords, then the value for maxReturns is set to false. That is, what is returned is not a full set.

Example — Input:

```
<GetUsersListParameters maxRecords="50">
    <filters>
        <surnamefilter value="D*"/>
        <givennamefilter value="James"/>
        <initialsfilter value = "S"/>
        <usertypefilter value="ALL"/
    </filters>
    <organizedBy entity="LAST_NAME"/>
</GetUsersListParameters>
```

The output returned depends on the number of search entries. If the search entries exceed a configured limit in the Management Services or a limit stipulated in the directory source, then user accounts are broken into buckets. You can use GetUsersList repeatedly, substituting or replacing the returned filters along with additional filters to get a limited set of users.

As an alternative, each user element is returned along with a masterid, which uniquely identifies the user in the Scalix world, and can be used to query and obtain further or detailed information for modifying/deleting attributes.

Example — Output:

```
<ReturnGetUsersListParameters maxReturned="true">
    <user name="Bart Simpson" id="bart_simpson_master_id" classtype="Full
| Limited" mailnode="a,b" server="palermo.us.scalix.com" usertype="M"/>
    ....
</ReturnGetUsesListParameters>
```

GetUserInfo

Type — Stateless

Use this remote method to fetch user-specific information. For example: Given a user's masterid, get me additional information.

The method is used primarily by the Scalix Management Console for rendering and displaying user-related information that can be modified in subsequent operations.

Input parameter subschema:

```
<xsd:element name="GetUserInfoParameters"
type="GetUserInfoParametersType"/>
<xsd:complexType name="GetUserInfoParametersType">
    <xsd:attribute name="id" type="xsd:decimal" use="required"/>
</xsd:complexType>
```

Output parameter subschema:

```
<xsd:element name="ReturnGetUserInfoParameters"
type="ReturnGetUserInfoParametersType"/>

<xsd:complexType name="ReturnGetUserInfoParameters">
    <xsd:sequence>
        <xsd:element name="entitiy" type="xsd:string" minOccurs="0"
maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:attribute name="entityType"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="type" type ="xsd:enumType"/>
    <xsd:attribute name="master" type "xsd:masterEnumType"/>
</xsd:complexType>

<xsd:simpleType name="entityType">
    <xsd:attribute name="name" xsd:string"/>
```

```
    <xsd:attribute name="value" xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="enumType">
    <restriction base="string"
        <xsd:enumeration value="M"/>
        <xsd:enumeration value="I"/>
    </restriction>
</xsd:simpleType>

<xsd:simpleType name="masterEnumType">
    <restriction base="string"
        <xsd:enumeration value="L"/>
        <xsd:enumeration value="F"/>
    </restriction>
</xsd:simpleType>
```

Example — Input:

```
<GetUserInfoParameters id = "bart_simpson_master_id"/>
```

Example — Output:

```
<ReturnGetUserInfoParameters type="M" master="L">
    <entity name="CN" value="Bart Simpson"/>
    <entity name="S" value="Simpson"/>
    <entity name="G" value="Bart"/>
    <entity name="I"value="I"/>
    <entity name="INTERNET-ADDR" value="Bart
Simpson"<Bart.Simpson@gandalf.scalix.local"/>
    <entity name="INTERNET-ADDR" value="barts@gandalf.scalix.local"/>
    <entity name="HOME-PHONE" value="123-44-45678"/>
    <entity name="CNTRY" value="USA"/>
    <entity name="L"value="Thing City"/>
    <entity name="PD-OFFICE-NAME" value="San Mateo"/>
    <entity name="UL-AUTHID" value="biff.anders@KERBEROS_REALMS"/>
</ReturnGetUserInfoParameters>
```

GetExtraUserInfo

Type — Stateless

Use this remote method to fetch extra or additional user information that cannot be retrieved via the GetUserInfo() function. For instance, to get account status (locked or unlocked), LOGGEDIN times, and service level, use this method. The default values, if missing, for CAN_USE_SWA and SENDER are true.

The syntax is similar to GetUserInfo.

Example — Input:

```
<GetExtraUserInfoParameters id = "bart_simpson_master_id"/>
```

Example — Output:

```
<ReturnExtraUserInfoParameters>
```

```
    <entity name="ACCOUNT_STATUS"value="unlocked"/>
    <entity name="LAST_SIGNON"value="04.14.05 02:06:28"/>
    <entity name="SERVICE_LEVEL" value="0"/>
    <entity name="SENDER" value="false"/>
    <entity name="CAN_USE_SWA" value="false"/>
    <entity name="SIS_URL" value="sis://xxx.yuy..."/>
    <entity name="RECOVERY_FOLDER_VISIBLE" value="false"/>
</ReturnExtraUserInfoParameters>
```

GetUserLoginStatus

Type — Stateless

Use this method to determine whether a particular user is currently logged in. There is no need to supply <ScalixServers> element.

The syntax similar to GetUserInfo.

Example — Input:

```
<GetUserLoginStatusParameters id="guid"/>
```

Example — Output:

```
<ReturnUserLoginStatusParameters>
   <entity name="LOGGED_IN" value="true | false | unknown"/>
</ReturnUserLoginStatusParameters>
```

The value “unknown” indicates it could not tell if the user is logged in, possibly because of some connection problems.

AddUser

Type — Stateless

Use this method to add new users to a specific or targeted remote server. The user element with attribute type determines whether the user is a local Scalix user (MAIL) or an Internet user (INTERNET). If the user type is INTERNET, then the subsequent command to be executed is omaddent. That is, it does not have a mailbox on any of the Scalix servers. As such, the Management Services elect which Scalix server to add a directory entry to for the user if no mailNode and server is specified. Otherwise, that is dictated by the mailNode and the ScalixServer element in the SOAP message.

Input parameter subschema:

```
<xsd:element name="AddUserParameters" type="AddUserParametersType"/>

<xsd:complexType name="AddUserParametersType">
    <xsd:sequence>
        <xsd:element name="user" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute="type" type="xsd:mailUserEnum"
use=required"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="mailNode" type="xsd:string" minOccurs="0">
```

```
            <xsd:complexType>
                <xsd:attribute="name" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="userAttributes" type="userAttributesType"
minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name=userAttributesType">
    <xsd:sequence>
        <xsd:element name="attribute" type="entityType"
maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="mailUserEnumType">
    <restriction base="string">
        <xsd:enumeration value="MAIL"/>
        <xsd:enumeration value="INTERNET"/>
    </restriction>
</xsd:simpleType>
```

Output parameter subschema:

```
<xsd:element name="ReturnAddUserParameters" type="ReturnAddUserParametersType"/>

<complexType name=ReturnAddUserParameters">
    <xsd:sequence>
        <xsd:element name="user" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="id" type="xsd:decimal"/>
</xsd:complexType>
```

Example – Input:

```
<AddUserParameters>
    <user type="MAIL"/>
    <mailNode name="gandalf,scalix"/>
    <userAttributes>
        <entity name="S" value = "Anderson"/>
        <entity name="G"value ="Biff"/>
        <entity name="I"value="I"/>
        <entity name="INTERNET-ADDR" value="biff.anderson@simpson"/>
        <entity name="PASSWORD" value="1@mduhBOss"/>
        <entity name="UL-AUTHID" value="biff.anders@KERBEROS_REALMS"/>
        <entity name="ADMIN" value="true"/>
        <entity name="MBOXADMIN" value = "true"/>
        ....
    </userAttributes>
</AddUserParameters>
```

Example — Output:

```
<ReturnAddUserParameters id="biff_anderson_master_id"/>
```

ModifyUser

Type — Stateless

Use this method to modify the attributes of a particular user just created. The masterid uniquely identifies the user in the Scalix system. Through the masterid, the Management Services determine which Scalix server owns the directory entry, and then directs its request for modifications. The Management Services query LDAP with the masterid type of user. If the Scalix user has a mailbox, then it issues ommodu. Otherwise, it resorts to ommodent. For Scalix users only, user attributes such as PHONE1- or MOBILE-PHONE, which cannot be modified by ommodu, are modified by ommodent.

Input parameter subschema:

```
<xsd:element name="ModifyUserParameters" type="ModifyUserParametersType"/>

<complexType name="ModifyUserParametersType">
    <xsd:sequence>
        <xsd:element name="userAttributes" type="userAttributesType "/>
    </xsd:sequence>
    <xsd:attribute name = "id" type="xsd:decimal" use="required"/>
</complexType>
```

Output parameter subschema:

```
<xsd:element name="ReturnModifyUserParameters"
type="ReturnModifyUserParametersType"/>

<complexType name=ReturnModifyUserParameters">
    <xsd:sequence>
        <xsd:element name="user" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="id" type="xsd:decimal"/>
</xsd:complexType>
```

Example — Input:

The userAttributes are the ones with new values. In the example here, if the user is a Scalix user, then ommodu is used to modify CN, S, G, I, UL-AUTHID, ADMIN, whereas ommodent is used to modify the PHONE-1 attribute. In the case of Internet users or users provisioned outside the Scalix world, all user attributes are modified via the ommodent.

```
<ModifyUserParameters id="biff_anderson_master_id"/>
    <userAttributes>
        <entity name="CN" value = "Biff S. Bothersome"/>
        <entity name="S" value="Bothersome"/>
        <entity name="G" value="Biff"/>
        <entity name="I" value="S"/>
        <entity name="UL-AUTHID" value="biff.anders@KERBEROS_REALMS"/>
        <entity name="ADMIN" value = "true"/>
        <entity name="PHONE-1" value="123456789"/>
```

```
    </userAttributes>
</ModifyUserParameters>
```

Example — Output:

```
<ReturnModifyUserParameters/>
```

DeleteUser

Type — Stateless

This method deletes a user from the appropriate remote server given the ID. An empty return parameter list is returned for a successful operation. Otherwise, a fault message is generated and sent back.

Input parameter subschema:

```
<xsd:element name="DeleteUserParameters" type="DeleteUserParametersType"/>

<complexType name=DeleteUserParameters">
    <xsd:attribute name="id" type="xsd:decimal" use="required"/>
</complexType>
```

Output parameter subschema:

```
<xsd:element name="ReturnDeleteUserParameters"
type="ReturnDeleteUserParametersType"/>

<complexType name=ReturnDeleteUserParametersType"/>
```

Example — Input:

```
<DeleteUserParameters id="bill_anderson_master_id"/>
```

Example — Output:

```
<ReturnDeleteUserParameters/>
```

AddResource

Type — Stateless

Use this method to create or add a new resource on a specific mailnode. Examples are conference rooms, projectors, and printers. The USERLIST and SYSTEM attributes are a subset of those provided for AddUser and ModifyUser methods. What follows here in the example is the full list currently supported by AddResource.

The syntax similar to AddUser.

Example — Input:

```
<AddResourceParameters>
    <mailNode name="palermo,us"/>
    <resourceAttributes>
         <entity name="S" value="Room"/>
         <entity name="CN" value="Board Room"/>
         <entity name="INTERNET-ADDR" value="xyz@scalix.com"/>
```

```
            <entity name="UL-AUTHID" value="Board Room"/>
            <entity name="PASSWORD" value="secret"/>
            <entity name="RECURRENCE" value="true"/>
            <entity name="CAN_CONFLICT" value="true"/>
            <entity name="PHONE-1" value="112233445555"/>
            <entity name="PHONE-2" value="222333444544"/>
            <entity name="FAX" value="33344455666"/>
            <entity name="CNTRY" value="US"/>
            <entity name="L" value="San Mateo"/>
            <entity name="ENTRY-DESC" value="Scalix Board Room"/>
            <entity name="EMPL-DEPT value="HR"/>
            <entity name="EX-CDA-DIRECTORY" value="E"/>
            <entity name="UL-CLASS" value="Full"/>
            <entity name="UL-IL value="French"/>
        </resourceAttributes>
    </AddResourceParameters>
```

Example — Output:

The client can use additional items returned, such as console, to display additional items on mouseover events.

```
<ReturnAddResourceParameters id="resource_id" "phone-1="112233445555"
phone-2="222333444544" fax="33344455666" city="San Mateo"/>
```

ModifyResource

Type — Stateless

Use this method to alter any of the attributes listed in the previous section for AddResource. The underlying Management Services split them into the omaddu and ommodent calls, respectively.

The syntax is similar to ModifyUser.

Example — Input:

```
<ModifyResourceParameters id="resource_id"/>
    <resourceAttributes>
        <entity name="RECURRENCE" value="false"/>
        <entity name="FAX" value="44567890"/>
        <entity name="CAN_CONFLICT" value="false"/>
    </resourceAttributes>
</ModifyResourceParameters>
```

Example — Output:

```
<ReturnModifyResourceParameters/>
```

DeleteResource

Type — Stateless

This method removes the resource with the given ID from the Scalix directory on the Scalix server.

The syntax is similar to DeleteUser.

Example — Input:

```
<DeleteResourceParameters id="resource_id"/>
```

Example — Output:

```
<ReturnDeleteResourceParameters/>
```

GetResourceInfo

Type — Stateless

This method fetches all information about a particular resource given its ID.

The syntax is similar to GetUserInfo.

Example — Input:

```
<GetResourceParameters id="resource_id"/>
```

Example — Output:

```
<ReturnGetResourceInfoParameters type="M" master="L">
    <entity name="S" value="Room"/>
    <entity name="CN"  value="Board Room"/>
    <entity name="INTERNET-ADDR"  value="xyz@scalix.com"/>
    <entity name="UL-AUTHID"value="Board Room"/>
    <entity name="RECURRENCE" value="true"/>
    <entity name="CAN_CONFLICT" value="true"/>
    <entity name="PHONE-1"value="112233445555"/>
    <entity name="PHONE-1" value="222333444544"/>
    <entity name="FAX" value="33344455666"/>
    <entity name="CNTRY" value="US"/>
    <entity name="L" value="San Mateo"/>
    <entity name="ENTRY-DESC" value="Scalix Board Room"/>
    <entity name="EMPL-DEPT value="HR"/>
    <entity name="EX-CDA-DIRECTORY" value="E"/>
    <entity name="UL-CLASS" value="Full"/>
    <entity name="HOST-FQDN" value="palermo.us.scalix.com"/>
    <entity name="MAILNODE" value="palermo,us"/>
    <entity name="GLOBAL-UNIQUE-ID value ="guid"/>
    <entity name="LOCAL-UNIQUE-ID value="lguid"/>
</ReturnGetResourceInfoParameters>
```

GetResourcesList

This method fetches all resources across all servers managed by Scalix. The filters supported in this method are cnfilter, surnamefilter, mailnodefilter, serverfilter, and emailfilter. The resourcetypefilter is of type FULL or LIMITED.

The syntax is similar to GetUsersList.

Example — Input:

```
<GetResourcesListParameters maxRecords="50">
    <filters>
        <cnfilter value="D*"/>
        <serverfilter>value="palermo.us.scalix.com"/>
        <resourcetypefilter> value="FULL|LIMITED"/>
        <mailnodefilter value="palermo,us"/>
    </filters>
</GetResourcesListParameters>
```

Example — Output:

```
<ReturnGetResourcesListParameters maxReturned="true">
    <resource name="Board Room-123" id="guid", phone-1="12345566" phone-
2="45678889 fax="456778990" city="San Mateo" mailnode="a,b"
server="palermo.us.scalix.com" classtype="Full |  Limited"/>
    <resource name="Engineering Room-345 id="guid" phone-1="5677889"
phone-2="67856789" fax="89876587" city="New York" mailnode="a,b"
server="palermo.us.scalix.com" classtype="Full |  Limited"/>
    ....
</ReturnGetResourcesListParameters>
```

Public Directory List (Group) Functions

Like the methods for user provisioning in the directory, these methods primarily deal with the Scalix directory. Public distribution lists (PDLs) are entries like any other user in the directory. When you add a group to Scalix an e-mail address is created for that group and any members of the group receive the e-mail. So if you create a group called sales, all members of group receive e-mail addressed to sales@yourcompany.com. As such, groups are also called PDLs. PDLs are added to this directory, and then after that, members who already exist in the directory can be added to a PDL.

How information is manipulated is transparent to the client. It is done through the interface defined in this section. The functions are stateless, which means that no record is kept of preceding action so any request must contain all information required.

The basic public directory list (group) functions you can access via the API are:

- AddGroup
- GetGroupsList
- GetGroupInfo
- ModifyGroup
- DeleteGroup
- AddMembersToGroup
- DeleteMembersFromGroup
- AddMembersToGroupAccess
- DeleteMembersFromGroupAccess
- ModifyMembersGroupAccess
- GetGroupAccessMemberList
- GetMemberAccessGroupList

Each is outlined here.

AddGroup

Type — Stateless

This method creates a new PDL on the target remote Scalix server. It returns its newly-unique ID. Because PDLs can have no mailbox associated with them, the list can be added to any Scalix server or the one specified in the SOAP message. If the type is MAIL, then the PDL goes on the server with the specified “mailNode”.

Input parameter subschema:

```
<xsd:element name="AddGroupParameters" type="AddGroupParametersType"/>

<xsd:complexType name="AddGroupParametersType">
    <xsd:sequence>
        <xsd:element name="user" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute="type" type="xsd:mailUserEnum"
use=required"/
```

```
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="mailNode" type="xsd:string" minOccurs="0">
            <xsd:complexType>
                <xsd:attribute="name" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="userAttributes" type="userAttributesType"
minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name=userAttributesType">
    <xsd:sequence>
        <xsd:element name="attribute" type="entityType" minOccurs="0",
maxOccurs="unbounded”/>
    </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="mailUserEnumType">
    <restriction base="string">
        <xsd:enumeration value="MAIL"/>
        <xsd:enumeration value="INTERNET"/>
    </restriction>
</xsd:simpleType>
```

Output parameter subschema:

```
<xsd:element name="ReturnAddGroupParameters" type="ReturnAddGroupParametersType"/>

<complexType name=ReturnAddGroupParameters">
    <xsd:sequence>
        <xsd:attribute="id" type="xsd:decimal "/>
    </xsd:sequence>
</xsd:complexType>
```

Example – Input:

```
<AddGroupParameters>
   <user type="MAIL"/>
     <mailNode="pippin,scalix"/>
      <userAttributes>
          <entity name="CN" value="Admin   Users"/>
              ....
   ....
      </userAttributes>
</AddGroupParameters>
```

Example – Output:

```
<ReturnAddGroupParameters id = "admin_users_master_id"/>
```

GetGroupsList

Type — Stateless

This method is similar in function and syntax to GetUsersList, except that it does not support OR filters and you cannot use mailnodefilter in conjunction with containsfilter.

Input parameter subschema:

```
<xsd:element name="GetGroupsListParameters" type="GetGroupsListParametersType"/>

<xsd:complexType name="GetGroupsListParametersType">
    <xsd:sequence>
        <xsd:elementname="filters" type="filtersType"/>
        <xsd:element name="organizedBy" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="entity" type="groupFilterEnumType"/>
            </xsd:complexType>
        </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="filtersType">
    <xsd:sequence>
        <xsd:element name="namefilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="mailnodefilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="serverfilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="mailfilter" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="containsfilter" type="xsd:string"
minOccurs="0">
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="typefilter" type="xsd:string" minOccurs="0">
            <xsd:complexType>
                <xsd:attribute name="value" type="groupType/>
            </xsd:complexType>
        </xsd:element>
```

```
    </xsd:sequence>
</xsd:complexType>

<!-- simple type enum definitions -->
<xsd:simpleType name="groupType">
    <restriction base="string">
        <xsd:enumeration value="admin"/>
        <xsd:enumeration value="normal"/>
    </restriction>
</xsd:simpleType>

<xsd:simpleType name="groupFilterEnumType">
    <restriction base="string">
        <xsd:enumeration value="NAME"/>
    </restriction>
</xsd:simpleType>
```

Output parameter subschema:

```
<xsd:element name="ReturnGetGroupsListParameters"
type="ReturnGetGroupsListParametersType"/>

<xsd:complexType name="ReturnGetGroupsListParametersType"/>
    <xsd:sequence>
        <xsd:element name="group" type="xsd:string" minOccurs="0",
maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:attribute="name" type="xsd:string"/>
                <xsd:attribute="id" type="xsd:string"/>
                <xsd:attribute="server" type=:xsd:string"/>
                <xsd:attribute="mailnode" type= "xsd:string"/>
                <xsd:attribute="master" type="xsd:string">
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
```

Example — Input:

Get a list of groups that satisfy a search filter constraint. The namefilter is the name of the Group; typefilter can be used to retrieve Administrative Groups or Normal groups. Absence of a typefilter returns all groups.

```
<GetGroupsListParameters>
    <filters>
         <namefilter value="D*"/>
         <containsfilter value="andy_palay_id”/>
    </filters>
    <organizedBy entity = "NAME"/>
</GetGroupsListParameters>
```

or

```
<GetGroupsListParameters>
    <filters>
```

```
        <namefilter value="D*"/>
        <mailnodefilter value="admins,scalix"/>
    </filters>
    <organizedBy entity = "NAME"/>
</GetGroupsListParameters>
```

or use the filter to fetch only administrative groups:

```
<GetGroupsListParameters>
    <filters>
        <typefilter value="admin"/>
    </filters>
</GetGroupsListParameters>
```

or use the filter to fetch all groups minus the administrative groups:

```
<GetGroupsListParameters>
    <filters>
        <typefilter value="normal"/>
    </filters>
<GetGroupsListParameters>
```

Example — Output:

Returns a set of PDLs that satisfy the search criteria with a name and id attributes.

```
<ReturnGetGroupsListParameters >
    <group name="Admin Users" id="admin_users_master_id" master="L"
server="verona.scalix.local" mailnode="verona, scalix"/>
    <group name="Marketing" id = “marketing_id_master_id" master="F"
server="milano.scalix.local" mailnode="milano,scalix"/>
    ....
    <group name="Web Developers" id = “web_developers_master_id"
master="L" server="verona.scalix.local" mailnode="verona,scalix"/>
</ReturnGetGroupsListParameters>
```

GetGroupInfo

Type — Stateless

This method returns all members of the PDL identified by the unique ID. Note that groups can contain other groups that are identified by the attribute type.

Input parameter subschema:

```
<xsd:element name="GetGroupInfoParameters"
type="GetGroupInfoParametersType"/>

<xsd:complexType name="GetGroupInfoParametersType"
    <xsd:attribute name="id" type="xsd:decimal" use="required"/>
</xsd:complexType>
```

Output parameter subschema:

```
<xsd:element name="ReturnGetGroupInfoParameters"
type="ReturnGetGroupInfoParametersType"/>

<xsd:complexType name="ReturnGetGroupInfoParameters">
    <xsd:sequence>
        <xsd:element name="member" type="xsd:string"xsd:attribute
name="name" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
                <xsd:attribute name="id" type="xsd:decimal"/>
                <xsd:attribute name="type" type="enumMemberType"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="type" type ="xsd:enumType"/>
    <xsd:attribute name="master" type "xsd:masterEnumType"/>
</xsd:complexType>

<xsd:simpleType name="enumMemberType">
    <restriction base="string">
        <xsd:enumeration value="user"/>
        <xsd:enumeration value="group"/>
    </restriction>
</xsd:simpleType>

<xsd:simpleType name="enumType">
    <restriction base="string"
        <xsd:enumeration   value="M"/>
        <xsd:enumeration value="I"/>
    </restriction>
</xsd:simpleType>

<xsd:simpleType name="masterEnumType">
    <restriction base="string"
        <xsd:enumeration value="L"/>
        <xsd:enumeration value="F"/>
    </restriction>
</xsd:simpleType>
```

Example – Input:

```
<GetGroupInfoParameters id = "admin_users_master_id"/>
```

Example – Output:

```
<ReturnGetGroupInfoParameters type="M" master="L">
    <entity name="CN" value="List Name"/>
    <entity name="MAILNODE" value="boromir,scalix"/>
    <entity name="INTERNET-ADDR" value="List.Name@boromir.scalix.local"/>
    <member name=”Andy Palay” id=”andy_palay_master_id” type=”user”
usertype="M" classtype="Full | Limited"/>
    <member name=”Super Users” id=”super_users_master_id” type=”group"
usertype="M" classtype="Full | Limited"/>
</scalix:caa:ReturnGetGroupInfoParameters>
```

ModifyGroup

Type – Stateless

This method modifies the name of the group.

Input parameter subschema:

```
<xsd:element name="ModifyGroupParameters"
type="ModifyGroupParametersType"/>

<complexType name="ModifyGroupParametersType">
    <xsd:sequence>
        <xsd:element name="userAttributes" type="userAttributesType”/>
    </xsd:sequence>
    <xsd:attribute name = "id" type="xsd:decimal" use="required"/>
</complexType>
```

Output parameter subschema:

```
<xsd:element name="ReturnModifyGroupParameters"
type="ReturnModifyParametersType"/>

<complexType name=ReturnModifyParameters">
    <xsd:sequence>
        <xsd:element name="bucket" type="xsd:string"xsd:attribute
name="name" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="name" type="xsd:string"/>
                <xsd:attribute name="id" type="xsd:decimal"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
```

Example – Input:

```
<ModifyGroupParameters id="admin_users_master_id">
     <userAttributes>
         <entity name="CN" value = "New Name"/>
         ...
     </userAttributes>
</ModifyGroupParameters>
```

Example – Output:

```
<ReturnModifyGroupParameters/>
```

DeleteGroup

Type — Stateless

This method removes a PDL. The “fa” attribute is optional. This supports lookup by Lightweight Directory Access Protocol (LDAP) for foreign or exported groups from Active Directory, where it has no notion of a GUID.

Input parameter subschema:

```
<xsd:element name="DeleteGroupParameters"
type="DeleteGroupParametersType"/>

<complexType name=DeleteGroupParameters">
    <xsd:attribute name="id" type="xsd:decimal" use="required"/>
    <xsd:attribute name="fa" type="xsd:string" use="optional"/
</complexType>
```

Output parameter subschema:

```
<xsd:element name="ReturnDeleteGroupParameters"
type="ReturnDeleteGroupParametersType"/>

<complexType name=ReturnDeleteGroupParametersType"/>
```

Example — Input:

```
<DeleteGroupParameters id="admin_users_master_id"/>
```

or

```
<DeleteGroupParameters fa="dn_foreign_address"/>
```

Example — Output:

```
<ReturnDeleteGroupParameters/>
```

AddMembersToGroup

Type — Stateless

This method allows the client to add members to the PDL. Members are identified by their master ID.

Input parameter subschema:

```
<xsd:element name="AddMembersToParameters"
type="AddMembersToParametersType"/>

<xsd:complexType name="AddMembersToGroupParametersType">
    <xsd:sequence>
        <xsd:element name="member" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute="id" type="xsd:decimal" use=required/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
```

Example — Input:

```
<AddMembersToGroupParameters id=”3456”>
    <member id=”john_doe_master_id”>
    <member id=”managers_master_id”/>
</AddMembersToGroupParameters>
```

Example — Output:

```
<ReturnAddMembersToGroupParameters/>
```

DeleteMembersFromGroup

Type — Stateless

This method allows the client to delete members from a PDL.

Input parameter subschema:

```
<xsd:element name="DeleteMembersFromParameters"
type="DeleteMembersFromParametersType"/>

<xsd:complexType name="DeleteMembersFromGroupParametersType">
    <xsd:sequence>
        <xsd:element name="member" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute="id" type="xsd:decimal" use=required/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
```

Example — Input:

```
<DeleteMembersFromGroupParameters id=”3456”>
    <member id=”john_doe_master_id”/>
    <member id=”managers_master_id”/>
</DeleteMembersFromGroupParameters>
```

Example — Output:

```
<ReturnDeleteMembersFromGroupParameters/>
```

AddMembersToGroupAccess

Type — Stateless

Use this method to add members who can manage this group with specified capabilities. This is the equivalent of using the ACI command line for PDLs or groups. Access capabilities can be “read, modify, remove, and config”. The request executes on the computer that owns the group. The request does not require the <ScalixServer> element.

Input parameter subschema:

```
<xsd:element name="AddMembersToGroupAccessParameters"
type="AddMembersToGroupAccessParametersType"/>

<xsd:complexType name="AddMembersToGroupAccessParametersType">
    <xsd:sequence>
        <xsd:element name="member" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute="id" type="xsd:decimal" use=required/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:sequence>
         <xsd:element name="name" type="accessEnumType">
             <xsd:complexType>
                 <xsd:attribute="value" type="xsd:string" use=required/>
             </xsd:complexType>
         </xsd:element>
    <xsd:sequence>

    </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="accessEnumType">
    <restriction base="string"
        <xsd:enumeration value="modify"/>
        <xsd:enumeration value="read"/>
        <xsd:enumeration value="config"/>
        <xsd:enumeration value="remove"/>
    </restriction>
</xsd:simpleType>
```

Example — Input:

```
<AddMembersToGroupAccessParameters id = "group_guid">
    <members id="guid"/>
    <members id="guid"/>
    ...
    <members id="guid">
    <accessAttributes>
        <entity name = "modify" value="true"/>
        <entity name = "read"  value="true"/>
        <entity name = "config" value="true"/>
        <entity name = "remove" value="true"/>
    </accessAttributes>
</AddMembersToGroupAccessParameters>
```

Example — Output:

```
<ReturnAddMembersToGroupAccessParameters/>
```

DeleteMembersFromGroupAccess

Type — Stateless

Use this method to delete member access from a particular group. This deletes or removes their access to manage or administer the group. The method or request executes on the computer that owns the group. The method ignores the <ScalixServer> element.

The syntax is similar to AddMembersToGroupAccess.

Example — Input:

```
<DeleteMembersFromGroupAccessParameters id = "group_guid">
    <member id="guid"/>
    <member id="guid"/>
    ...
    <member id="guid">
</DeleteMembersFromGroupAccessParameters>
```

Example — Output:

```
<ReturnMembersFromGroupAccessParameters/>
```

ModifyMembersGroupAccess

Type — Stateless

Use this method to modify existing access privileges for members from a particular group. The method or request executes on the computer that owns the group. The method ignores the <ScalixServer> element.

The syntax is similar to AddMembersToGroupAccess.

Example — Input:

```
<ModifyMembersGroupAccessParameters id="group_id">
    <member id="guid"/>
    <member id="guid"/>
    <accessAttributes>
        <entity name = "modify" value="false"/>
        <entity name = "config" value="false"/>
        <entity name = "remove" value="true"/>
    </accessAttributes>
</ModifyMembersGroupAccessParameters>
```

Example — Output:

```
<ReturnModifyMembersGroupAccessParameters/>
```

GetGroupAccessMemberList

Type — Stateless

Use this method to retrieve a list of all members with access privileges for a particular or specified group. The method or request executes on the computer that owns the group. The method ignores the <ScalixServer> element.

Example — Input:

```
<GetGroupAccessMemberListParameters id="group_id"/>
```

Example — Output:

```
<ReturnGroupAccessMemberListParameters>
    <member name="Danny Tran" id="guid" modify="true" read="true"
config="true" usertype="M" classtype="Full | Limited"/>
    <member name="Jules Damji" id="guid" modify="true" read="true"
usertype="M" classtype="Full | Limited"/>
    ...
    <member name="Andy Palay" id="guid" modify="true" read="true"/>
</ReturnGroupAccessMemberListParameters>
```

GetMemberAccessGroupList

Type — Stateless

Use this method to retrieve a list of all groups for which the specified user identified by its GUID has specified access privileges. If the <ScalixServer> element is not specified, this method executes on all registered servers. The group element attribute server=“fqdn” is the name of the server on which the group resides for which the user has the requested privileges.

The accessAttributes specified here must exist or have been set to “true” by AddMembersToGroupAccess.

Example — Input

```
<GetMemberAccessGroupListParameters id="member_guid">
    <accessAttributes>
        <entity name = "modify"/>
        <entity name = "read"/>
    </accessAttributes>
</GetMemberAccessGroupListParameters>
```

Example — Output:

```
<ReturnGetMemberAccessGroupListParameters>
    <group name="GroupA" id="guid" server="verona.scalix.local/>
    <group name="GroupB" id="guid" server="milano.scalix.local"/>
</ReturnGetMemberAccessGroupListParameters>
```

Login/Logout Functions

The console issues these functions to log in and log out. Optionally, other clients can use the same request to log in if they want.

The basic login and logout functions you can access from the API are:

- Login
- Logout

Login

Type — Stateful

Provides the login request to the Management Services. These credentials are used by the Management Services to bind to LDAP and to confirm whether the user has “admin” privileges. It also uses the same credentials to obtain the Kerberos Service Ticket for the Remote Execution Service.

Input parameter subschema:

Empty

Output parameter subschema:

```
<xsd:element name="ReturnLoginParameters"
type="ReturnLoginParametersType"/>

<complexType name=ReturnLoginParametersType">
    <xsd:sequence>
        <xsd:element name="access" type="xsd:enumAccessType"/>
    </xsd:sequence>
</xsd:complexType>
```

Output parameter subschema:

```
<xsd:element name="ReturnLoginParameters"
type="ReturnLoginParametersType"/>

<complexType name=ReturnLoginParametersType">
    <xsd:sequence>
        <xsd:element name="access" type="xsd:enumAccessType"/>
        <xsd:element name="priviledge" type="xsd:attribute name="type"
type="xsd:string>
            <xsd:complexType>
                <xsd:attribute  name="value" type="xsd:string"
maxOccurs="unbounded"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="ReturnLoginParameters"
type="ReturnLoginParametersType"/>

<complexType name=ReturnLoginParametersType">
```

```
    <xsd:sequence>
        <xsd:element name="access" type="xsd:enumAccessType"/>
        <xsd:element name="priviledge" type="xsd:attribute name="type"
type="xsd:string>
            <xsd:complexType>
                <xsd:attribute name="value" type="xsd:string"
maxOccurs="unbounded"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
```

Example – Input:

```
<LoginParameters/>
```

Example – Output:

```
<ReturnLoginParameters>
    <access value="granted"/>
    <entity name="CN" value="List Name"/>
    <entity   name="INTERNET-ADDR"
value="List.Name@boromir.scalix.local"/>
    <entity name="GLOBAL-UNIQUE-ID" value="guid"/>
    <priviledge type ="addUser" value = "1"/>
    <priviledge type ="deleteUser" value = "1"/>
    <priviledge type ="modifyUser" value   = "1"/>
    <priviledge type   ="modifyUserAttributes" value = "1"/>
    <priviledge type ="addGroup" value =   "1"/>
    <priviledge type ="deleteGroup"   value = "1"/>
    <priviledge type ="modifyGroup" value =   "1"/>
    <priviledge type ="modifyGroupMembers" value = "1"/>
    <priviledge type ="modifyServerSettings" value = "1"/>
    <priviledge type ="administerServers" value = "1"/>
</ReturnLoginParameters>
```

Or if access is not granted, then the following is returned:

```
<ReturnLoginParameters>
    <access value="denied">
    <priviledge/>
</ReturnLoginParameters>
```

Logout

Type — Stateless

The console issues this command to log out from the Management Services manager. All user sessions and data associated with it are deleted.

Input parameter subschema:

```
<xsd:element name="LogoutParameters" type="LoginParametersType"/>

<complexType name=LogoutParameters</complexType>
```

Output parameter subschema:

```
<xsd:element name="ReturnLogoutParameters"
type="ReturnLoginParametersType"/>

<complexType name=ReturnLogoutParametersType"/>
```

Example — Input:

```
<LogoutParameters/>
```

Example — Output:

```
<ReturnLogoutParameters/>
```

Server Functions

The basic server functions you can access from the API are:

- StartServer
- StopServer
- StartService
- StopService
- GetServicesList
- GetActiveUsersList
- GetServiceInfo
- GetServersList
- GetServerInfo
- GetServerEventLog
- ModifyService
- GetServerLicenses
- GetServerLicenseFeaturesList
- AddServerLicenseFeaturesList
- DeleteServerLicense
- GetPluginInfo
- RunPlugin
- GetPluginsList
- GetServerNameGenerationRules
- ModifyServerNameGenerationRules
- AddServerMailAddressGenerationRules
- GetServerMailAddressGenerationRules
- ModifyServerMailAddressGenerationRules
- DeleteServerMailAddressGenerationRules
- GetServerGeneralSettings
- ModifyServerGeneralSettings
- DeleteServerGeneralSettings
- GetUserGeneralSettings
- ModifyUserGeneralSettings
- DeleteUserGeneralSettings

Each is explained here.

StartServer

Type — Stateless

This method allows any remote server to be started. If <ScalixServers> in the SOAP message is empty, then all registered Scalix servers are started. To specify a single server, use <ScalixServers><Host><FQDN></Host></ScalixServers>.

Input parameter subschema:

```
<StartServerParameters/>
```

Output parameter subschema:

```
<xsd:element name="ReturnStartServerParameters"
type="ReturnStartServerParametersType"/>

<complexType name="ReturnStartServersParameters">
    <xsd:sequence>
        <xsd:element name="server" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="name" type="xsd:string"/>
                <xsd:attribute name="status"type="xsd:string />
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</complexType>
```

Example — Input:

```
<StartServerParameters/>
```

Example — Output:

```
<ReturnStartServerParameters>
    <server name="hostname.scalix.local" status="On"/>
    <server name="sting.scalix.local" status="On"/>
</ReturnStartServerParameters>
```

StopServer

Type — Stateless

This method allows any remote server to be stopped. If <ScalixServers> in the SOAP message is empty, then all registered Scalix server are stopped. To specify a single server, use <ScalixServers><Host><FQDN></Host></ScalixServers>.

Example — Input:

```
<StopServerParameters/>
```

Example — Output:

```
<xsd:element name="ReturnStopServerParameters"
type="ReturnStopServerParametersType"/>

<complexType   name="ReturnStopServerParameters">
```

```
    <xsd:sequence>
        <xsd:element name="server" type="xsd:string">
            <xsd:complexType>
                <xsd:attribute name="name" type="xsd:string"/>
                <xsd:attribute name="status" type="xsd:string />
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</complexType>
```

Example — Output:

```
<ReturnStopServerParameters>
   <server name="hostname.scalix.local" status="Off"/>
   <server name="sting.scalix.local"status="Off"/>
</ReturnStopServerParameters>
```

StartService

Type — Stateless

This method allows any remote server service/daemon to be started. If <ScalixServers> in the SOAP message is empty, then all registered Scalix servers' specified services or daemons are started. To specify a single server, use ScalixServers><Host><FQDN></Host></ScalixServers>

Input parameter subschema:

```
<xsd:element name="StartServiceParameters" type="xsd:string">
    <xsd:complexType>
        <xsd:attribute name="service" type="serviceEnumType"/>
    </complexType>
    <xsd:simpleType name="serviceEnumType">
        <restriction base="string"
            <xsd:enumeration value="ldap"/>
            <xsd:enumeration value="pop"/>
            <xsd:enumeration value="lci"/>
            <xsd:enumeration value="rci"/>
            <xsd:enumeration value="applink"/>
            <xsd:enumeration value="cda"/>
            <xsd:enumeration value="local"/>
            <xsd:enumeration value="omscan"/>
            <xsd:enumeration value="print"/>
            <xsd:enumeration value="request"/>
            <xsd:enumeration value="router"/>
            <xsd:enumeration value="test"/>
            <xsd:enumeration value="omdbmon"/>
            <xsd:enumeration value="drs"/>
            <xsd:enumeration value="imap"/>
            <xsd:enumeration value="iss"/>
            <xsd:enumeration value="mime"/>
            <xsd:enumeration value="ns"/>
            <xsd:enumeration value="nnd"/>
            <xsd:enumeration value="pop"/>
        </restriction>
```

```
   </xsd:simpleType>
</xsd:element>
```

Example — Input:

```
<StartServiceParameters service="ldap"/>
```

Example — Output:

```
<ReturnStartServiceParameters>
    <server name="hostname.scalix.local" service = "ldap" status="On"/>
    <server name="sting.scalix.local"service = "ldap" status="On"/>
</ReturnStartServiceParameters>
```

StopService

Type — Stateless

This method stops any remote server service/daemon. If the value <ScalixServers> in the SOAP message is empty, then all registered Scalix servers' specified service or daemon stop. To specify a single server, use <ScalixServers><Host><FQDN></Host></ScalixServers>

Input parameter subschema:

```
<xsd:element name="StopServiceListParameters" type="xsd:string">
    <xsd:complexType>
        <xsd:attribute name="service" type="serviceEnumType"/>
    </complexType>
    <xsd:simpleType name="serviceEnumType">
        <restriction base="string"
            <xsd:enumeration value="ldap"/>
            <xsd:enumeration value="pop"/>
            <xsd:enumeration value="lci"/>
            <xsd:enumeration value="rci"/>
            <xsd:enumeration value="applink"/>
            <xsd:enumeration value="cda"/>
            <xsd:enumeration value="local"/>
            <xsd:enumeration value="omscan"/>
            <xsd:enumeration value="print"/>
            <xsd:enumeration value="request"/>
            <xsd:enumeration value="router"/>
            <xsd:enumeration value="test"/>
            <xsd:enumeration value="omdbmon"/>
            <xsd:enumeration value="drs"/>
            <xsd:enumeration value="imap"/>
            <xsd:enumeration value="iss"/>
            <xsd:enumeration value="mime"/>
            <xsd:enumeration value="ns"/>
            <xsd:enumeration value="smtpd"/>
            <xsd:enumeration value="pop"/>
        </restriction>
    </xsd:simpleType>
</xsd:element>
```

Example — Input:

```
<StopServiceParameters service="dirsync"/>
```

Example — Output:

```
<ReturnStopServiceParameters>
    <server name="hostname.scalix.local" service = "dirsync"
status="Off"/>
    <server name="sting.scalix.local"service = "dirsync" status="Off"/>
    .....
</ReturnStopServiceParameters>
```

GetServicesList

Type — Stateless

Use this method to get a list of all services/daemons and the current states of Scalix servers managed by the Management Services. If no server is specified, it gets information from all the registered servers.

Example — Input:

```
<GetServicesListParameters/>
```

Example — Output:

Returns the list of Scalix services and daemons running on each registered server with the Management Services.

```
<ReturnGetServicesListParameters>
    <server name="verona.scalix.local">
        <service name ="ldap" desc="LDAP Daemon" status="On"
time="16:58:57" numOfUsers="-1"/>
        <service name= "router" desc="Service Router" status="Off"
time="17:29:52" numOfUsers="23"/>
        ...
        <service name="smd" desc="Shared Memory Daemon" status="Always"
time="" numOfUsers="-1"/>
    </server>
</ReturnGetServicesListParameters>
```

GetActiveUsersList

Type — Stateless

Use this method to get a list of all users currently logged in on a particular Scalix server. Currently, all clients go through the Remote Client Interface (rci) service.

The syntax for server arguments is similar to GetServicesList.

Example — Input:

```
<GetActiveUsersListParameters/>
```

Example Output:

```
<ReturnActiveUsersListParameters>
    <server name="verona.scalix.local"/>
        <user service="rci" CN="Jules Damji" mailnode="verona/scalix"
pid="24597" loginTime="15:14:37" client="Unlicensed Client"/>
        <user service="rci" CN=sxadmin" mailnode="verona/scalix" pid =
26077" loginTime="15:39:42" client="SWAClient"/>
        <user service="rci" CN=sxadmin" mailnode="verona/scalix" pid =
26062" loginTime="15:39:36" client="SWAClient"/>
        <user service="rci" CN=sxadmin" mailnode="verona/scalix" pid =
26077" loginTime="15:39:36" client="SWAClient"/>
    <server/>
</ReturnActiveUsersListParameters>
```

GetServiceInfo

Type — Stateless

Use this method to get detailed information about the service. You specify the server name in the SOAP message as part of the ScalixServer element.

Example — Input:

```
<GetServiceInfoParameters name="router"/>
```

Example — Output:

```
<ReturnGetServiceInfoParameters>
    <entity name="SUBSYSTEM" value = "Service Router"/>
    <entity name="SERVICE_NUMBER value="2"/>
    <entity name="NUMBER_OF_COMPONENTS" value="2"/>
    <entity name="LOGGING_LEVEL" value="7"/>
    <entity name="AUDIT_LEVEL" value="0"/>
    <entity name="HAS_INPUT_QUEUE" value="YES"/>
    <entity name="QUEUE_NAME" value="ROUTER"/>
    <entity name="DISPLAY_IN_OMSTAT" value="YES"/>
    <entity name="CAN_BE_ENABLED" value="YES"/>
    <entity name="REQUIRED_STATE" value="Enabled"/>
    <entity name="LAST_STATE_CHANGE" value="11.15.04"/>
    <entity name="LAST_DELAYED_OFF_TIME" value="11.15.04"/>
    <entity name="START_PROGRAM" value="~/bin/service.router"/>
    <entity name="STOP_PROGRAM" value="~/bin/shut.queue -q ROUTER -d %d -
s 2"/>
    <entity name="STATUS_PROGRAM" value=""/>
    <entity name="COMPONENT_PIDS" value="11518 11519"/>
    <entity name="NICE_VALUE" value="1"/>
    <entity name="RESOLVE_FLAG" value="0"/>
    <entity name="CONTROLLED_BY_ALL" value="YES"/>
    <entity name="MIN_TEMP_PROCESSES" value="0"/>
    <entity name="MAX_TEMP_PROCESSES" value="0"/>
    <entity name="MAX_AUX_PROCESSES" value="0"/>
    <entity name="CONTEXT_INFO" value="0"/>
    <entity name="AUXILLARY_PIDS" value=""/>
    <entity name="CONTEXT_INFO" value = ""/>
</ReturnGetServiceInfoParameters/>
```

GetServersList

Type — Stateless

Use this method to get a list of servers managed by the Management Services.

Example — Input:

```
<GetServersListParameters/>
```

Example — Output:

Returns the list of Scalix servers that have registered themselves to the Management Services.

```
<ReturnGetServersListParameters >
    <server name="hostname.scalix.local"/>
    <server name="pippin.scalix.local"/>
</ReturnGetServersListParameters>
```

GetServerInfo

Type — Stateless

Use this method to get information about what Scalix components are installed on the Scalix Server. This method must have as part of the SOAP message:
<ScalixServers><Host>server.name</Host></ScalixServers>

Example — Input:

No arguments to this method. Because this is a server operation, the directed server must be part of the ScalixServer element.

```
<GetServerInfoParameters/>
```

Example — Output:

Only components installed on this server appear in the XML, and those can be “sac”, “swa”, “res”, or “server”.

```
<ReturnGetServerInfoParameters server="verona.scalix.local">
    <component name = "scalix-sac" version=""9.1.0.173" installed="Mon 18
Oct 2004 05:09:31 PM PDT" size=10105929" release="1"/>
   <component name = "scalix-server" version="9.2.0.15" installed="Fri 05
Nov 2004 04:07:29 PM PST" size="71531876" release="alpha.fc2"/>
    <component name = "scalix-swa" version="9.1.0.158" installed="Thu 30
Sep 2004 10:41:09 AM PDT" size="6351250" release="1"/>
</ReturnGetServerInfoParameters>
```

GetServerEventLog

Type — Stateless

Use this method to get the event log for any Scalix server. This method must have a <ScalixServers> element. Only a single server can be specified.

Example — Input:

service — Name of the service for which event logging is required. If absent, it returns all events.

level — Show entries made at or below this level. The level numbers represent the level of detail of the log entries, and are interpreted as:

1 Show only SERIOUS ERRORS.

3 Show ERRORS and SERIOUS ERRORS.

5 Show WARNINGS, ERRORS, and SERIOUS ERRORS.

7 Show REPORTS of successful execution of commands, WARNINGS, ERRORS, and SERIOUS ERRORS.

9 Show REPORTS from standard mailing processes, other REPORTS, WARNINGS, ERRORS, and SERIOUS ERRORS.

Note that Event Log entries have been configured to be LOGGEDIN by SetServerEventLogLevel.

For example, if the logging level is set to 5 for a service with omconflvl, no additional levels can be displayed for that service by setting omshowlog at a level higher than 5.

fromdate — Show entries made on or after this date

fromtime — Show entries made at or after this time. If absent it will be assumed 00:00.

todate — Show entries made on or before this date. If absent then today's date is used.

totime — Show entries made at or before this time

```
<GetServerEventLogParameters maxLimit="5000"/>
    <filters>
        <service value="admin"/>
        <levelvalue="7"/>
        <fromdatevalue="dd.mm.yy"/>
        <fromtimevalue="hh:mm"/>
        <todatevalue="dd.mm.yy"/>
        <totimevalue="hh:mm"/>
    </filters>
</GetServerEventLogParameters maxLimit>
```

Example — Output:

```
<ReturnGetServerEventLogParameters maxLimitExceeded="false">
    <line value="log line entry..."/>
    <line value="log line entry2..."/>
    <line value="log line entry3.."/>
    ...
    <line value="log line entryN.."/>
</ReturnGetServerEventLogParameters/>
```

ModifyService

Type — Stateless

Use this method to modify the event logging level for a particular service on all or some Scalix servers. If no <ScalixServers> element is specified, this operation executes on all Scalix managed servers.

Example — Input:

```
<ModifySerivceParameters name="router">
    <serviceAttributes>
        <entiry name="level value="9"/>
    </serviceAttributes>
</ModifyServiceParameters>
```

Example — Output:

```
<ReturnModifyServiceParameters/>
```

GetServerLicenses

Type — Stateless

Use this method to get all licenses from all registered servers. If no <ScalixServers> element is specified, this operation is executed on all servers managed by the Management Services.

Example — Input:

```
<GetServerLicensesParameters/>
```

Example — Output:

```
<ReturnGetServerLicensesParameters>
    <server name="palermo.us.scalix.com">
    <license type="permanent" body="
--Scalix Licence Key
License Type: Permanent
System Class: Multi-Server
Domain: scalix.com
Domain: bamail.net
LVID: 2020-12-31
Enterprise Users: 1000
Community Users: unlimited
--Scalix Licence Signature
dm+//gbazBjkngBFcRrNKkKxCPwG7NzxuQnEe7OK0RpBQZRG1n7Bv3AZgA84m7p5tNUrTxh9
1QWuxbvIiw9m3opkMIUTwHTcSZF4oGxPf2FcFHAQ7VG608byH/aDjJnr7flF1//8l7luGEn4
bZO/SDXe7xdTiK/5UdpGH6wozcU=
--Scalix Licence End"/>
    <license type="temporary" body="contents_of_the_license_file_here"/>
    <server>
    <server name="milano.us.scalix.com">
        <license type="permanent"
body="contents_of_the_license_file_here"/>
        <license type="temporary"
body="contents_of_the_license_file_here"/>
    </server>
```

```
</ReturnGetServerLicenseParameters>
```

GetServerLicenseFeaturesList

Type — Stateless

Use this method to retrieve a list of license features. If no <ScalixServer> element is specified, it fetches license features from each registered server. They all should be the same.

Example — Input

```
<GetServerLicenseFeaturesListParameters/>
```

Example — Output

```
<ReturnGetServerLicenseFeaturesListParameters>
    <server name="palermo.us.scalix.com">
        <feature type="RECOVERY_FOLDER"/>
        <feature type="HIGH_AVAILABILITY"/>
        <feature type="MIGRATION"/>
        <feature type="MULTI_INSTANCE"/>
        <feature type="MULTI_SERVER"/>
        <feature type="RECOVERY_FOLDER"/>
        <feature type="TNEF_GATEWAY"/>
        <feature type="WIRELESS"/>
    </server>
</ReturnGetServerLicenseFeaturesListParameters>
```

AddServerLicenseFeaturesList

Type — Stateless

Use this method to add or update a license. This operation is global. That is, the operation or method applies across all managed servers. No <ScalixServer> element is required.

Example — Input:

```
<AddServerLicenseParameters>
    <licenseAttributes>
        <entity name="type" value="permanent" | "temporary"/>
        <entity name="body" value="contents_of_the_license_file_here"/>
    </licenseAttributes>
</AddServerLicenseParameters>
```

Example — Output:

```
<ReturnAddServerLicenseParameters/>
```

DeleteServerLicense

Type — Stateless

Use this method to delete a server license. This operation is global. That is, the operation applies across all managed servers. No <ScalixServer> element is required.

Example — Input:

```
<DeleteServerLicenseParameters type="permanent" | "temporary"/>
```

Example — Output:

```
</ReturnDeleteServerLicenseParameters>
```

GetPluginInfo

Type — Stateless

This method returns specific information regarding a plug-in specified by server and name. The <ScalixServers> element must be specified.

Example — Input:

```
<GetPluginInfoParameters name = “addtestusers”/>
```

Example — Output:

```
<ReturnGetPluginInfoParamerers>
    <entity name="Name" value=“Add Test Users”/>
    <entity name="Description" value=“Adds a specified number of test
users”/>
    <entity name=”OutputType" value=“text/plain”/>
    <entity  name="Version" value="1.0:"/>
    <parameters>
        <parameter flag=“-n” name=”Number of users” value=”10”
valueType=”int” argType=”single” description=”Number of test users”/>
       <parameter flag=“-s” name=”User Names” value=”” valueType=”string”
argType=”multi” description=”Any names provided will be used for test
users. Additionally user names will be generated automatically”/>
    </parameters>
</ReturnGetPluginInfoParamerers>
```

RunPlugin

Type — Stateless

This method returns the output from running a plug-in specified by server and name with the option of any parameters supplied. When no <ScalixServers> element is specified, this operation is executed on all Scalix-managed servers.

Example — Input:

```
<RunPluginParameters name="addtestusers">
    <parameters>
        <parameter flag="-n" value="10"/>
        <parameter flag="-s" value="Jules"/>
        <parameter flag="-s" value="Sasha"/>
        <parameter flag="-s" value="Anneke"/>
    </parameters>
</RunPluginParameters name>
```

Example — Output:

```
<ReturnRunPluginParameters>
    <server name="verona.scalix.local">
        <entity name="results" value="users added successfully"/>
    </server>
</ReturnRunPluginParameters>
```

GetPluginsList

Type — Stateless

Use this method to get plug-ins from registered servers for which the user specified in the authid parameter has execute permissions. If no authid parameter is provided, the call returns all plug-ins for which the user specified with the authentication credentials has execute permissions. If no <ScalixServers> element is specified, this operation is executed on all Scalix servers managed by the Management Services.

Example — Input:

```
<GetPluginsListParameters authid="sasha@scalix.com"/>
```

Example — Output:

```
<ReturnGetPluginsListParameters>
    <server name="palermo.us.scalix.com">
        <Plugin name="checkqueues"/>
        <Plugin name="monitordisk"/>
        <Plugin name="addtestusers"/>
    </server>
    <server name="milano.us.scalix.com">
        <Plugin name="addtestusers"/>
        <Plugin name="docollate_entries"/>
    </server>
</ReturnGetPluginsListParameters>
```

GetServerNameGenerationRules

Type — Stateless

Use this method to retrieve server name generation rules. Those include: Common Name, Internet Generation rules, and Authentincation ID rules. When no server is specified, it retrieves from all registered servers. See the table “Name Generation Rule Attributes” on page 112 for the rule names and their respective supported format values.

Example — Input:

```
<GetServerNameGenerationRulesParameters/>
```

Example — Output:

```
<ReturnGetServerNameGenerationRulesParameters>
   <server name="verona.scalix.local">
       <entity name="general.usrl_cn_rule" value="G I. S"/>
       <entity name="general.usrl_authid_rule" value="G_I_S"/>
       <entity name="general.inet_domain_rule" value="scalix.com"/>
   </server>
   <server name="milano.scalix.local">
       <entity name="general.usrl_cn_rule" value="S,G.I"/>
       <entity name="general.usrl_authid_rule" value="G_I_S"/>
       <entity name="general.inet_domain_rule" value="scalix.local"/>
   </server>
</ReturnGetServerNameGenerationRulesParameters>
```

or

```
<ReturnGetServerNameGenerationRulesParameters/>
```

ModifyServerNameGenerationRules

Type — Stateless

Use this method to modify server name generation rules. If no server is specified, it sets those settings for all registered servers.

Example — Input:

```
<ModifyServerNameGenerationRulesParameters>
    <ruleAttributes>
        <entity name="general.usrl_authid_rule" value="G_I_S"/>
    </ruleAttributes>
</ModifyServerNameGenerationRulesParameters>
```

Example — Output:

```
<ReturnModifyServerNameGenerationRulesParameters/>
```

AddServerMailAddressGenerationRules

Type — Stateless

Use this method to add server system Internet address generation rules. When no server is specified, it adds to all registered servers. You can create a maximum of five generation rules.

Example — Input:

```
<AddServerMailAddressGenerationRulesParameters>
    <ruleAttributes>
        <rule name_part="giS" domain_part="scalix.com" slot="1"/>
        <rule name_part="gis" domain_part="scalix.local" slot="2"/>
        <rule name_part="G.I.S" domain_part=bamail.net" slot="3"/>
    </ruleAttributes>
</AddServerMailAddressGenerationRulesParameters>
```

Example — Output:

```
<ReturnAddServerMaillAddressGenerationRules/>
```

GetServerMailAddressGenerationRules

Type — Stateless

Use this method to retrieve server system Internet address generation rules. When no server is specified, it fetches information from all registered servers.

Example — Input:

```
<GetServerMailAddressGenerationRulesParameters/>
```

Example — Output:

```
<ReturnServerMailAddressGenerationRulesParameters>
    <server name="verona.scalix.local">
        <rule name_part="giS" domain_part="scalix.com" slot="1"/>
        <rule name_part="gis" domain_part="scalix.local" slot="2"/>
        <rule name_part="G.I.S" domain_part=bamail.net" slot="3"/>
    </server>
</ReturnServerMailAddressGenerationRulesParameters>
```

ModifyServerMailAddressGenerationRules

Type — Stateless

Use this method to modify server Internet address generation rules. When no server is specified, it modifies all registered servers. If the name_part is " ", then it removes or deletes that rule, which is an alternative way to delete a rule associated with a specific slot number.

Example — Input:

```
<ModifyServerMailAddressGenerationRulesParameters>
    <ruleAttributes>
        <rule name_part="gis" domain_part="scalix.local" slot="2"/>
        <rule name_part="" domain_part="" slot="3"/>
    </ruleAttributes>
</ModifyServerMailAddressGenerationRulesParameters>
```

Example — Output:

```
<ReturnModifyServerMailAddressGenerationRulesParameters/>
```

DeleteServerMailAddressGenerationRules

Type — Stateless

Use this method to delete server Internet address generation rules. When no server is specified, it deletes from all registered servers.

Example — Input:

```
<DeleteServerMailAddressGenerationRulesParameters>
    <ruleAttributes>
        <rule slot="4"/>
        <rule slot="5"/>
    </ruleAttributes>
</DeteletServerMailAddressGenerationRulesParameters>
```

Example — Output:

```
<ReturnDeteletServerMailAddressGenerationRulesParameters/>
```

GetServerGeneralSettings

Use this method to retrieve Server configuration settings from the server general.cfg file. When no server is specified, it returns all settings from the general.cfg file. If no <configAttributes> is specified, then all attributes currently set in the general.cfg file are set back. Otherwise, only the requested values of the settings are sent back. See the table “Server General Settings” on page 113 for names of friendly tags to use for retrieving individual or a selected set of parameters.

Example — Input:

```
<GetServerGeneralSettingsParameters/>
```

or

```
<GetServerGeneralSettingsParameters>
```

```
    <configAttributes>
        <entity name="general.IMAP_CONNECTION_LIMIT"/>
        <entity name="general.IMAP_IDLE_TIMEOUT"/>
    </configAttributes>
<GetServerGeneralSettingsParameters>
```

Example — Output:

```
<ReturnGetServerGeneralSettingsParameters>
    <entity name="general.ual_signon_alias" value="YES"/>
    <entity name="general.ual_config_alias" value="SYS"/>
    <entity name="general.ual_use_alias"value="FALSE"/>
    <entity name="general.cda_use_changelog" value="TRUE"/>
    <entity name="general.IMAP_CONNECTION_LIMIT" value="500"/>
    <entity name="general.IMAP_CONNRATE_LIMIT" value="10"/>
    <entity name="general.IMAP_IDLE_TIMEOUT"value="31"/>
    <entity name="general.LD_CREATE_MESSAGE_STORE" value="TRUE"/>
    <entity name="general.DS_CUST_SEND_REQ_NOW"value="TRUE"/>
    <entity name="general.DS_CUST_PERIOD_TIMER_MINUTES" value="TRUE"/>
    ...
</ReturnGetServerGeneralSettingsParameters>
```

or if the <configAttributes> element is present, then the output only contains values for those parameters:

```
<ReturnGetServerGeneralSettingsParameters>
    <entity name="general.IMAP_CONNRATE_LIMIT" value="10"/>
    <entity name="general.IMAP_IDLE_TIMEOUT" value="31"/>
</ReturnGetServerGeneralSettingsParameters>
```

ModifyServerGeneralSettings

Type — Stateless

Use this method to modify Scalix server settings in the general.cfg file. If no server element is specified in the <ScalixServers>, this operation affects each registered Scalix server. See the table “Server General Settings” on page 113 for tags to specify in the <configAtttibutes> element.

Example — Input:

```
<ModifyServerGeneralSettingsParameters>
    <configAttributes>
        <entity name="general.IMAP_CONNRATE_LIMIT" value="10"/>
        <entity name="general.IMAP_IDLE_TIMEOUT" value="31"/>
    </configAttributes>
</ReturnModifyServerGeneralSettingsParameters>
```

Example — Output:

```
<ReturnModifyServerGeneralSettingsParameters/>
```

DeleteServerGeneralSettings

Type — Stateless

Use this method to remove or delete a Scalix server in the general.cfg file. If no server element is specified in <ScalixServers>, then this operation impacts each registered Scalix server. Use the table “Server General Settings” on page 113 for the tags to specify in the <configAtttibutes> element.

Use this method only if necessary.

Example — Input:

```
<DeleteServerGeneralSettingsParameters>
    <configAttributes>
        <entity name="general.cda_use_changelog">
    </configAttributes>
</DeleteServerGeneralSettingsParameters>
```

Example — Output:

```
<ReturnServerGeneralSettingsParameters/>
```

GetUserGeneralSettings

Type — Stateless

Use this method to retrieve user configuration settings from the users.cfg file. If <configAttributes> is absent, then all attributes currently set in the user.cfg file are retrieved, otherwise only the requested values of the settings are sent back.

Example — Input:

```
<GetUserGeneralSettingsParameters id="user_guid"/>
```

or

```
<GetUserGeneralSettingsParameters id="user_guid">
    <configAttributes>
        <entity name="user.MAPI_MBC_ALLOWED"/>
    </configAttributes>
<GetUserGeneralSettingsParameters>
```

Example — Output:

```
<ReturngGetUserGeneralSettingsParameters>
    <entity name="user.MAPI_MBC_ALLOWED" value="TRUE"/>
    ...
</ReturnGetUserGeneralSettingsParameters>
```

ModifyUserGeneralSettings

Type — Stateless

Use this method to modify user settings in the users.cfg file.

Modify only after reading the documentation and fully understanding its effects.

Example — Input:

```
<ModifyUserGeneralSettingsParameters id="user_guid">
    <configAttributes>
        <entity name="user.IMAP_CONNRATE_LIMIT" value="10"/>
        <entity name="user.MAPI_MBC_ALLOWED" value="FALSE"/>
    </configAttributes>
</ReturnModifyUserGeneralSettingsParameters>
```

Example — Output:

```
<ReturnModifyUserGeneralSettingsParameters/>
```

DeleteUserGeneralSettings

Type — Stateless

Use this method to remove or delete settings in the users.cfg file.

Use this method only if necessary.

Example — Input:

```
<DeleteUserGeneralSettingsParameters id="user_guid">
    <configAttributes>
        <entity name="user.MAPI_MBC_ALLOWED"/>
    </configAttributes>
</DeleteUserGeneralSettingsParameters>
```

Example — Output:

```
</ReturnUserGeneralSettingsParameters>
```

Queue Functions

The basic queue functions you can access from the API are:

- GetQueuesNameList
- GetQueuesList
- GetQueueInfo
- GetQueueMessagesList
- GetQueueMessageInfo
- DeleteMessageFromQueue

Each is explained here.

GetQueuesNameList

Type — Stateless

Use this method to get a list of Scalix queue names and their descriptions. There is no need to specify the Server target. It reads this information from the Management Services configuration files. Use these queue names for SOAP methods pertaining to Scalix queues.

Example — Input:

```
<GetQueuesNameListParameters/>
```

Example — Output:

Returns the list of all Scalix queue names and their descriptions. This is a memory dump of the Scalix queue table (“Scalix Queues” on page 111).

```
<ReturnGetQueuesNameListParameters>
    <queue name="LOCAL"  desc="Local Delivery Service input queue"/>
    <queue name="ROUTER" desc="Router Delivery input queue"/>
    ...
</ReturnGetQueuesNameListParameters>
```

GetQueuesList

Type — Stateless

Use this method to get a list of Scalix queues and the number of messages associated with them, along with load averages. If no server is specified, it retrieves information from all registered servers.

The syntax is similar to GetServicesList.

Example — Input:

```
<GetQueuesList/>
```

Example — Output:

The attributes have the following meanings:

name — Name of the Scalix queue

attachedMsgs — Number of current messages on the queue waiting to be processed

msgProcessedNow — Current number of messages being processed

totalMessages — Total number of message attached to this queue since the queue manager last started. It resets if the queue manager service restarts, and it also resets the load averages as well.

load1Min — One-minute load average of the queue

load5Min — Five-minute load average of the queue

load15Min — 15-minute load average of the queue

```
<ReturnGetQueuesListParameters>
    <server name="verona.scalix.local>
        <queue name="ARCHERR "attachedMsgs="10" msgsProcessedNow="2"
totalMessages=42 load1Min=5.0 load5Min=5.6 load15Min=10.1/>
        <queue name="ROUTER "attachedMsgs="10" msgsProcessedNow="2"
totalMessagesProcessed="42" load1Min="5.0"
load5Min="5.6"load15Min="10.1"/>
        ....
     <server/>
</ReturnGetQueuesListParameters>
```

GetQueueInfo

Type — Stateless

Use this method to retrieve specific queue information from a particular server. The server argument must be specified in the SOAP message.

Example — Input:

```
<GetQueueInfoParameters name="ROUTER"/>
```

Example — Output:

```
<ReturnQueueInfoParameters>
    <queue name="ROUTER "attachedMsgs="10" msgsProcessedNow="2"
totalMessagesProcessed="42" load1Min="5.0" load5Min="5.6"
load15Min="10.0"/>
</ReturnQueueInfoParameters>
```

GetQueueMessagesList

Type — Stateless

Use this method to obtain a list of current messages on a particular Scalix Server or all registered servers for a particular server queue. When no server argument is specified in the SOAP message as part of the <ScalixServers/> element, it gets messages for the specific QUEUE from all of them. Note that the data is transient and can change as messages are picked up for processing.

Example — Input:

```
<GetQueueMessagesListParameters name="SMERR"/>
```

Example — Output:

The type attribute can be the following enumeration:

type = MSG (Ordinary Message)

type = ACK (Acknowledgement)

type = REP (Reply)

The priority attribute can be the following enumerations:

priority= L (Low)

priority= N (Normal)

priority= U (Urgent)

```
<ReturnGetQueueMessagesList>
    <server name= "verona.scalix.local"/>
        <message msgNo="0" msgRefNo="25265" sender="+DIRSYNC" type="MSG"
priority="L" subject="DIRECTORY SYNCHRONIZATION - REQUEST_Updates"
sent="05.09.04"/>
        <message msgNo="1" msgRefNo="25249" sender="+DIRSYNC" type="MSG"
priority="L" subject="DIRECTORY SYNCHRONIZATION - REQUEST_Updates"
sent="05.09.04"/>
       ....
    </server>
</ReturnGetQueuesMessagesList>
```

GetQueueMessageInfo

Type — Stateless

Use this method to obtain message details from a specific Scalix Server on a particular or specified queue. You must specify the server argument in the SOAP message as part of the <ScalixServer/> element. Note that the data is transient and can be absent or already processed before the request reaches the server.

Example — Input:

```
<GetQueuesMessageInfoParameters msgRefNo="25265" name="SMERR"/>
```

Example — Output:

```
<ReturnQueueMessageInfoParameters>
    <entity name="REFNUM" value="25249"/>
    <entity name="MESSAGEID"
value="H000000000006281.1084146540.verona.scalix.local"/>
    <entity name="SENDER" value="+DIRSYNC / verona, scalix"/>
    <entity name="RECIPIENT" value="+DIRSYNC / MILANO, SCALIX"/>
    <entity name="RECIPIENT" value="+DIRSYNC / BUGS, SCALIX"/>
    <entity name="MESSAGETYPE" value="MSG"/>
    <entity  name="PRIORITY" value="L"/>
    <entity  name="SENT" value="5.09.04"/>
    <entity  name="ERRORTEXT" value="This is a MIME-encapsulated message"/
>
    <entity  name="ERRORTEXT" value="--i4A4hc221904.1084164279/
samwise.scalix.local"/>
    ....
    <entity name="ERRORTEXT" value="from samwise.scalix.local
(root@localhost)"/>
<ReturnQueueMessageInfoParameters/>
```

DeleteMessageFromQueue

Type — Stateless

Use this method to delete a message from a specified Scalix Server from a particular specified queue. You must specify the server argument in the SOAP message as part of the <ScalixServer/> element. Note that the data is transient and can be absent or already processed before the request reaches the server.

Example — Input:

```
<DeleteMessageFromQueue msgRefNo="25265" name="SMERR"/>
```

Example — Output:

```
<ReturnMessageFromQueue/>
```

Mailnode Functions

The mailnode functions you can access from the API are:

- GetMailNodesList
- GetMailNodeInfo
- AddMailNode
- DeleteMailNode
- ModifyMailNode

Each is explained here.

GetMailNodesList

Type — Stateless

Use this method to get a list of mailnodes on the servers managed by the Management Services.

Example — Input:

```
<GetMailNodesListParameters/>
```

Example — Output:

```
<ReturnMailNodesListParameters >
    <mailnode name="pippin,scalix" server = "pippin.scalix.local"
type="local" domain="scalix.com"/>
    <mailnode name="internet,foobar/scalix" server =
"pippin.scalix.local" type="mime"/>
    <mailnode name="tnef,scalix" server = "hostname.scalix.local"
type="tnef"/>
</ReturnMailNodesListParameters>
```

GetMailNodeInfo

Type — Stateless

Use this method to get detailed mailnode information from servers managed by the Services Manager. No server argument is needed because the mailnode name maps to the server.

Example — Input:

```
<GetMailNodeInfoParameters name="mktg"/>
```

Example — Output:

```
<ReturnGetMailNodeInfoParameters type="P">
    <entity name="SERVER" value="verona.scaix.local"/>
    <entity name="MAILNODE" value="mktg"/>
    <entity name="DOMAIN" value = "scalix.com"/>
    <entity name="RULE value="G.S"/>
</ReturnGetMailNodeInfoParameters>
```

AddMailNode

Type — Stateless

Use this method to create a mailnode on a particular server. Note that mailnodes have to be unique across a set of Scalix servers, so server name must be specified. If not, it picks the first registered server.

Rules can be of the following format:

G.S (Joe.Smith)

G.I.S(Joe.M.Smith)

gS(jSmith)

gs(js)

gis(jms)

giS(jmSmith)

The domain and rule are optional fields.

Example — Input:

```
<AddMailNodeParameters>
    <userAttributes>
        <entity name="MAILNODE" value="mktg"/>
        <entity name="DOMAIN" value = "scalix.com"/>
        <entity name="RULEvalue="G.S"/>
    </userAttributes>
</AddMailNodeParameters>
```

Example — Output:

```
<returnAddMailNodeParameters name="mktg"/>
```

DeleteMailNode

Type — Stateless

Use this method to delete a mailnode. There is no need for server arguments because mailnodes are unique, and the Management Services keep track of server-to-mailnode mappings so it directs the request to the appropriate server.

Mailnodes can only be deleted if there are no users on them.

Example — Input:

```
<DeleteMailNodeParameters name="mktg"/>
```

ModifyMailNode

Type — Stateless

Use this method to modify the domain associated with the mailnode and/or rule.

Changing the primary mailnode is not exposed at this interface.

Example — Input:

```
<ModifyMailNodeParameters name="mktg">
    <userAttributes>
        <entity name="DOMAIN" value="foo.com"/>
        <entity name="RULE" value="g.i.s"/>
    </userAttributes>
</ModifyMailNodeParameters>
```

Example — Output:

```
<ReturnModifyMailNodeParameters/>
```

Message Store or Mailbox Functions

The basic message store functions you can access from the API are:

- GetUserMessageStoreLimits
- ModifyUserMessageStoreLimits
- GetServerMessageStoreLimits
- ModifyServerMessageStoreLimits
- GetUserMessageStoreUsage
- GetServerMessageStoreUsage
- GetUsersMessageStoreUsageList
- DeleteUserMessageStoreItems
- CreateUserSmartCache
- CreateUserSISIndex

Each is outlined here.

GetUserMessageStoreLimits

Type — Stateless

Use this method to view user-specific message store limits. The limits are in KB. There is no need to specify the server because the Management Services can locate the user account, or determine which server owns the directory entry for the GUID.

Example — Input:

```
<GetUserMessageStoreLimits id="guid"/>
```

Example — Output:

The “limit” element attributes has the following meanings:

ms — Mailbox or message store limit in kilobytes. 0 means no limit.

it — In tray size in kilobytes. 0 means no limit.

fc — Filing cabinet size in kilobytes. 0 means no limit.

wb — Waste basket size in kilobytes. 0 means no limit.

pt — Pending tray size in kilobytes. 0 means no limit.

da — Distribution list area size in kilobytes. 0 means no limit.

The sanctions element attributes have the following meanings:

us — Causes a message to be sent to the user when one or more of the applicable limits are exceeded. The message indicates which limits have been exceeded.

as — Causes a message to be sent to the Error Notification User when a user exceeds one of their limits.

es — Causes a UAL error message to be generated when a user tries to create an item in a Filing Cabinet or Distribution List Area once it has exceeded the set limit. Note that this sanction does not apply to the message store limit.

rs — Causes delivery rejection when the user is exceeding limits. In this case, no messages are delivered to the user while they are exceeding their limits. Instead an NDN is returned to the originator of the message. Note that this sanction only applies to the message store limit.

ss — Enables messages sent by a user to be further processed by the deferred mail manager. For this option to have a useful effect, rules that perform some action based on the size of the user’s message store in relation to their configured limits must be configured. Typically, this option is used to reject any messages sent by the user after exceeding a limit.

If the attributes is missing, it has not been set. It only returns values for which the sanctions have been set.

```
<ReturnGetUserMessageStoreLimitsParameters>
    <entity name="ms" value="0"/>
    <entity name="it" value="500"/>
    <entity name="fc" value="10"/>
    <entity name="wb"value="10"/>
    <entity name="da" value="15"/>
    <entity name="us"value="true"/>
    <entity name="es" value="true"/>
</ReturnGetUserMessageStoreLimitsParameters>
```

ModifyUserMessageStoreLimits

Use this method to modify a user's specific message store limits. The limits are in KB. There is no need to specify the server because the Management Services can locate the user, or which server owns the directory entry for the GUID.

Example — Input:

```
<ModifyUserMessageStoreLimitsParameters id ="guid">
    <userAttributes>
        <entity name="ms" value="0"/>
        <entity name="it"value="500"/>
        ...
        <entity name="us"value="false"/>
        <entity name="es" value="true"/>
    </userAttributes>
</ModifyUserMessageStoreLimits>
```

Example — Output:

```
<ReturnModifyUserMessageStoreParameters/>
```

GetServerMessageStoreLimits

Type — Stateless

Use this method to get the message store limits along with sanction rules for each server or all servers. When no server name is specified in the <ScalixServers> element, it gets from all servers.

Example — Output:

```
<ReturnGetServerMessageStoreLimitsParameters>
    <server name="verona.scalix.local">
        <entity name="ms" value="0"/>
        <entity name="it" value="500"/>
        <entity name="fc" value="10"/>
        <entity name="wb" value="10"/>
        <entity name="da" value="15"/>
        <entity name="us" value="true"/>
        <entity name="es" value="true"/>
        <entity name="WARN_PCT" value="90"/>
        <entity name="WARN_TEXT" value="string"/>
        <entity name="REJECT_PCT" value="100"/>
    </server>
</ReturnGetServerMessageStoreLimitsParameters>
```

ModifyServerMessageStoreLimits

Type — Stateless

Use this method to modify global or server(s) message store limits along with sanctions if required. When no server is specified as part of the <ScalixServers> element, the operation applies across all servers.

Example — Input:

```
<ModifyServerMessageStoreLimitsParameters>
    <userAttributes>
        <entity name="ms" value="0"/>
        <entity name="it" value="500"/>
        ...
        <entity name="us" value="false"/>
        <entity name="es" value="true"/>
        <entity name="WARN_PCT" value="90"/>
        <entity name="WARN_TEXT" value="The desired warning message
string"/>
        <entity name="REJECT_PCT" value="100"/>
    </userAttributes>
</ModifyServerMessageStoreLimitsParameters>
```

Example — Output:

```
<ReturnModifyServerMessageStoreLimitsParameters/>
```

GetUserMessageStoreUsage

Type — Stateless

Use this method to retrieve user's message store size. There is no need to specify or indicate the <ScalixServers> element, because it will know what server to retrieve it from. All values returned are in KB.

Example — Input:

```
<GetUerMessageStoreUsageParameters id="guid"/>
```

Example — Output:

```
<ReturnGetUserMessageStoreUsageParameters>
    <entity name="MAILBOX" value="450"/>
    <entity name="WASTEBASKET" value="2"/>
</ReturnGetUserMessageStoreUsageParameters>
```

GetServerMessageStoreUsage

Type — Stateless

Use this method to retrieve the Scalix server's message store size. If no <ScalixServers> element is specified, it returns for each registered Scalix server. All values returned are in KB. Currently, only the total mailbox size and wastebasket folder or container is supported in the SOAP call. Along with the mailbox and wastebasket, the size of /var/opt/scalix is also returned (the parsed output of df -P -k /var/opt/scalix which is the file system path where the message store resides).

Example — Input:

```
<GetServerMessageStoreUsageParameters/>
```

Example — Output:

```
<ReturnGetServerMessageStoreUsageParameters>
    <server name="palermo.scalix.local">
        <entity name="MAILBOX"value="4500"/>
        <entity name="WASTEBASKET" value="26"/>
        <entity name="FILESYSTEM" value="74730664"/>
    </server>
    <server name="milano.scalix.local">
        <entity name="MAILBOX"value="4550"/>
        <entity name="WASTEBASKET" value="27"/>
        <entity name="FILESYSTEM" value="57135340"/>
    </server>
</ReturnGetServerMessageStoreUsageParameters>
```

GetUsersMessageStoreUsageList

Type — Stateless

Use this method to fetch Scalix servers' message store users list. This call can be used to retrieve the list of users who consume large amounts of disk space on the message store partition */var/opt/scalix*. Using various <usageAttributes> parameters, you can control what is returned from the server with respect to sorting order (ascending or descending), sorted by (mailbox or wastebasket), and maximum number of users. When no <ScalixServers> is specified, then it fetches information from all registered servers. It is advisable, for performance reasons, that you specify the <ScalixServers> element for the targeted server for which you want to fetch information.

The attributes values are as follows:

mb — Mailbox

wb — Wastebasket

all — Mailbox + wastebasket

ascend — Sort by ascending order

descend — Sort by descending order

Example — Input:

```
<GetUsersMessageStoreUsageListParameters>
    <usageAttributes>
```

```
        <entity name="SORT_ORDER" value = "ascend | descend"/>
        <entity name="SORT_BY" value = "mb | wb" | "all"/>
        <entity name="MAX_USERS" value = "10"/>
    </usageAttributes>
</GetUsersMessageStoreUsageListParameters>
```

When no <usageAttributes> element is specified in the SOAP call, then the defaults are SORT_ORDER is descend, SORT_BY is mb (mailbox) and MAX_USERS is 5.

Example — Output:

```
<ReturnGetUsersMessageStoreUsageListParameters>
    <server name="palermo.scalix.local">
        <user name="Julie Ferris" id="guid" mb="455" wb="55"/>
        <user name="Jules Damji" id="guid" mb="45" wb="5"/>
        <user name="Sasha Sterling id="guid" mb="43" wb="2"/>
        <user name="Andy Palay id="guid" mb="30" wb="1"/>
         ...
       /* If SORT_BY is "all" then the output will have the following user
entity line: */
        <user name="Julie Ferris" id="guid" total="610"/>
    </server>
</ReturnGetUsersMessageStoreUsageListParameters>
```

DeleteUserMessageStoreItems

Type — Stateless

Use this method to empty the user's message store, in particular to empty it of folders. Currently, only the wastebasket folder is supported. If no usage attributes are specified in the SOAP call, the default behavior empties the entire wastebasket, using no age or size criteria. AGE is in days and SIZE is in KB.

Example — Input:

```
<DeleteUserMessageStoreItemsParameters id="guid">
    <usageAttributes>
        <entity name="WASTEBASKET" value="true"/>
        <entity name="AGE"value="4"/>
        <entity name="SIZE"value="10"/>
    </usageAttributes>
</DeleteUserMessageStoreItemsParameters>
```

Example — Output:

```
<ReturnUserMessageStoreItemsParameters/>
```

CreateUserSmartCache

Type — Stateless

Use this method to initiate or prepare a user's SmartCache on the server where the user is provisioned. The max_cache_size_item is in MB. When no cacheAttributes is specified, the server defaults are used.

Example — Input:

```
<CreateUserSmartCacheParameters id="user_guid">
    <cacheAttributes>
        <entity name="CACHE_DIRECTORY"value="/
path_to_cache_directory_on_the_server"/>
        <entity name="MAX_CACHE_SIZE_ITEM  value="2"/> /*  default is
1000kb */
        <entity name="PASSWORD" value="user_password"/>
        /* only required if root is not executing this command */
    </cacheAttributes>
</CreateUserSmartCacheParameters>
```

or

```
<CreateUserSmartCacheParameters id="user_guid"/>
/* user all default server parameters; root must execute this */
```

Example — Output:

```
<ReturnCreateUserSmartCacheParameters/>
```

CreateUserSISIndex

Use this method to initiate or prepare a user's search index on the server where the user is provisioned. When no indexAttributes are specified, the defaults are used.

Example — Input:

```
<CreateUserSISIndexParameters id="user_guid">
   <indexAttributes>
       <entity name="SIS_URL" value="url"/>
       <entity name="TIME" value="days"/>
   </indexAttributes>
</CreateUserSISIndexParameters>
```

Example — Output:

```
<ReturnCreateUserSISIndexParameters/>
```

Password Controls

The basic password control functions you can access from the API are:

- GetPasswordSettings
- ModifyPasswordSettings

Each is explained here.

GetPasswordSettings

Type — Stateless

Use this method to set password controls globally or on a particular server. When no server is specified as part of the <ScalixServers> element, the operation or method applies across all registered servers.

Example — Input:

```
<GetPasswordSettingsParameters/>
```

Example — Output:

```
<ReturnGetPasswordSettingsParameters>
    <server name="verona.scalix.local">
        <entity name="EXPIRATION_DAYS" value="2"/>
        <entity name="MIN_LENGTH" value="8"/>
        <entity name="MAX_REPEAT_CHARS" value=""/>
```

or numeric value

```
        <entity name="MUST_HAVE_ALPHABETIC" value="true"/>
        <entity name="MUST_HAVE_LOWER_CASE" value="false"/>
        <entity name="MUST_HAVE_UPPER_CASE" value="true"/>
        <entity name="MUST_HAVE_NUMERIC" value="true"/>
        <entity name="MUST_HAVE_NON_ALPHANUMERIC" value="true"/>
        <entity name="MIN_REUSE_COUNT" value=""/>
        /* or numeric or number of times a password must change before a
previous value can be reused. */
        <entity name="MIN_REUSE_DURATION" value=""/>
        /* or numeric in days before old password can be reused */
        <entity name="MAX_LOGIN_RETRIES"value=""/>
        /* number of times before the user is locked out from illegal
password attempts. */
    </server>
</ReturnGetPasswordSettingsParameters>
```

ModifyPasswordSettings

Type — Stateless

Use this method to modify password controls globally or on a particular server. When no server is specified as part of the <ScalixServers> element, the operation or method applies across all registered servers.

Example — Input:

```
<ModifyPasswordSettingsParameters>
    <passwordAttributes>
        <entity name="EXPIRATION_DAYS" value="2"/>
        <entity name="MIN_LENGTH"value="8"/>
        <entity name="MAX_REPEAT_CHARS" value=""/>
        <entity name="MUST_HAVE_ALPHABETIC"value="true"/>
        <entity name="MUST_HAVE_LOWER_CASE"value="false"/>
        <entity name="MUST_HAVE_UPPER_CASE" value="true"/>
        <entity name="MUST_HAVE_NUMERIC" value="true"/>
        <entity name="MUST_HAVE_NON_ALPHANUMERIC" value="true"/>
        <entity name="MIN_REUSE_COUNT" value=""/>
        <entity name="MIN_REUSE_DURATION" value=""/>
        <entity name="MAX_LOGIN_RETRIES"value=""/>
    </passwordAttributes>
</ModifyPasswordSettingsParameters>
```

Note: To unset or reset password controls back to the default, use the method with only a single attribute: UNSET_SETTINGS. This attribute cannot be used with any other attributes.

```
<ModifyPasswordSettingsParameters>
    <passwordAttributes>
        <entity name="UNSET_SETTINGS value="true"/>
    </passwordAttributes>
</ModifyPasswordSettingsParameters>
```

Example — Output:

```
<ReturnModifyPasswordSettingsParameters/>
```

Management Services Settings

Console configurations along with Management Services configurations are stored in the ubermanager.properties file on the administration server.

Only the most relevant of the settings are exposed via the interface.

The Management Services settings you can access from the API are:

- GetConsoleConfig
- ModifyConsoleConfig
- GetDepartmentList
- AddDepartment
- DeleteDepartment
- ModifyDepartment

Each is explained here.

GetConsoleConfig

Type — Stateful. Configurations are stored in memory; any modifications are written to the disk in a file.

Use this method to retrieve Scalix Management Console configurations stored in the Management Services properties file.

Example — Input:

```
<GetConsoleConfigParameters/>
```

or

```
<GetConsoleConfigParameters>
    <configAttributes>
        <entity name="localDomain"/>
        <entity name="MaxListSize"/>
    </configAttributes>
</GetConsoleConfigParameters>
```

Example — Output:

LocalDomain is a multivalue attribute. For each value, an element for that entity is returned. When the <configAttributes> element with attributes is specified, only the requested elements are returned. Otherwise, all current settings are returned.

```
<ReturnGetConsoleConfigParameters>
    <entity name="LocalDomain" value="scalix.com"/>
    <entity name="LocalDomain" value="scalix.local"/>
    <entity name="ExternalAuthentication" value="false"/>
    <entity name="DefaultCountry" value="US"/>
    <entity name="DefaultCompany" value="Scalix Inc"/>
    <entity name="DefaultZipcode" value="94538"/>
    <entity name="AllowAuthenticationChoice" value="false"/>
    <entity name="MaxListSize" value="200"/>
</ReturnGetConsoleConfigParameters>
```

ModifyConsoleConfig

Type — Stateful. Configurations are stored in memory; any modifications are written to the disk in a file.

Use this method to modify Scalix Management Console settings. The modification is done in memory and reflected on the disk.

Example — Input:

LocalDomain is a multivalue attribute. For each value, an element for that entity is returned. To remove or add any of the multivalues, you must provide all the new values.

```
<ModifyConsoleConfigParameters>
    <configAttributes>
        <entity name="LocalDomain" value="scalix.net"/>
        <entity name="DefaultCountry" value="FR"/>
        <entity name="MaxListSize" value="250'/>
    </configAttributes>
</ModifyConsoleConfigParameters>
```

Example — Output:

```
<ReturnModifyConsoleConfigParameters/>
```

GetDepartmentList

Use this method to get a list of departments in the ubermanager.properties file.

Example — Input:

```
<GetDepartmentListParameters/>
```

Example — Output:

```
<ReturnDepartmentListParameters>
    <department name="Eng" id="Eng"/>
    <department name="Eng/SAC"id="Eng/SAC"/>
    <department name="Sales"id="Sales"/>
    <department name="Sales/US"id="Sales/US"/>
    <department name="Sales/UK"id="Sal,es/UK"/>
    <department name="Support"id="Support"/>
    <department name ="Marketing"id="Marketing"/>
    <department name="Marketing/Channels" id="Marketing/Channels"/>
    <department name="Marketing/Comm" id="Marketing/Comm"/>
</ReturnDepartmentListParameters>
```

AddDepartment

Use this method to add a department to an existing department list.

Example — Input:

```
<AddDepartmentParameters>
    <departmentAttributes>
        <entity name="NAME" value="Law"/>
    </departmentAttributes>
</AddDepartmentParemeters>
```

Example — Output:

```
<ReturnAddDepartmentParameters id="Law"/>
```

DeleteDepartment

Use this method to delete a department from an existing department list.

Example — Input:

```
<DeleteDepartmentParameters id="Law"/>
```

Example — Output:

```
<ReturnDeleteDepartmentParameters/>
```

ModifyDepartment

Use this method to modify a department from an existing department list.

Example — Input:

```
<ModifyDepartmentP:arameters id="Law"/>
    <departmentAttributes>
        <entity name="NAME" value="Legal"/>
    </departmentAttributes>
</ModifyDepartmentParameters>
```

Example — Output:

```
<ReturnModifyDepartmentParameters/>
```

Attributes Table for APIs

The table lists some of the attributes the client can specify in methods.

Table 1: Attributes

Name	LDAP Mapping	Methods	Type	Directory
S Surnname. For non-Scalix users, this only appears in the SYSTEM directory.	surname	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
S-TX Just like above but an 8-bit version.	N/A	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
G Given name. For non-Scalix users, this only appears in the SYSTEM directory.	givenName	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
G-TX Just like above but an 8-bit version.	N/A	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
I Initials. For non-Scalix users, this only appears in the SYSTEM directory.	initials	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
O X.400 attribute for organization.	o	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
O-TX Just like above but an 8-bit version.	N/A	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
I-TX Just like above but an 8-bit version.	N/A	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
CN Common name or display name. For non-Scalix users, this only appears in the SYSTEM directory.	cn, omCn	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
CN-TX Just like above but an 8-bit version.	N/A	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
Q Generation qualifier. Eg. James T. Kirk. Jr	generation-Qualifier (no hyphen)	AddUser, ModifyUser, GetUserInfo	String	USERLIST/SYSTEM
Q-TX Just like above but an 8-bit version.	N/A	AddUser, ModifyUser, GetUserInfo	String	USERLIST/SYSTEM

Table 1: Attributes

Name	LDAP Mapping	Methods	Type	Directory
INTERNET-ADDR Internet address or SMTP address. For non-Scalix users, this only appears in the SYSTEM directory.	omInterne-tAddr (no hyphen)	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
IA Short for Internet address above.	omInterne-tAddr (no hyphen)	AddUser, ModifyUser, AddGroup	String	USERLIST/SYSTEM
UL-AUTHID Login identity for Scalix users. It can be their Kerberos principal.	omUlAuthId	AddUser, ModifyUser	String	USERLIST
UL-IL Language setting for error mail messages.	omUlIl	AddUser, ModifyUser, GetUserInfo	String	USERLIST
EXTAUTH Externally authenticated, so passwords cannot be changed.	omUlcaps; bit 0x20	AddUser, ModifyUser	Boolean (true/ false)	USERLIST
[OU1-OU4] Mailnodes. For non-Scalix users, this only appears in the SYSTEM directory.	omMailNode	N/A	String	USERLIST/SYSTEM
[OU1-TX-OU4-TX] Just like above but an 8-bit version. Currently not supported in the Console.	N/A	N/A	String	USERLIST/SYSTEM
ADMIN This is not an X.400 attribute. It is a flag for the command line mapped into UL-CAP with appropriate bits set to indicate admin privileges.	omUlCaps	AddUser, ModifyUser, GetUserInfo	Boolean (true/ false)	USERLIST
MBOXADMIN This is not an X.400 attribute. It is a flag, just like above, that indicates mailbox administrative privileges.	omUlCaps	AddUser, ModifyUser, GetUserInfo	Boolean (true/ false)	USERLIST
PASSWORD This is not an X.400 attribute. It is flag just like above that converts the password in MD5 digests and stores into the USERLIST for Scalix users.	N/A	AddUser, ModifyUser, GetUserInfo	String	USERLIST

Table 1: Attributes

Name	LDAP Mapping	Methods	Type	Directory
PASSWORD-RESET This is not an X.400 attribute. It is flag just like above for the command line to indicate to the Scalix server that the password is to be reset upon login by the Scalix user.	N/A	AddUser, ModifyUser	String	None
EX-CDA-DIRECTORY CDA will exclude for publishing and indexing SYSTEM directory entries if the value is specified as a single char “E” or any non-null value.	omExCdaDirectory (no hyphen)	AddUser, ModifyUser, GetUserInfo	Char (“E”)	SYSTEM
PD-OFFICE-NAME Attribute for office name.		AddUser, ModifyUser	String	SYSTEM
POSTAL-CODE Attribute for postal code or zip code.		AddUser, ModifyUser	String	SYSTEM
POSTAL-ADDRESS-STR Attribute for postal street address.		AddUser, ModifyUser	String	SYSTEM
STATE-OR-PROVINCE X.400 attribute for state or province.	st	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
STREET-ADDRESS X.400 attribute for street address.	street, streetAddress	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
CNTRY X.400 attribute for country.	c	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
L X.400 attribute for city.	l	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
HOME-PHONE X.400 attribute for home telephone number.	homeTelephoneNumber (no hyphen)	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
PAGER-PHONE Attribute for pager telephone number.	pagerTelephoneNumber (no hyphen)	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
MOBILE-PHONE Attribute for mobile telephone number.	mobileTelephoneNumber (no hyphen)	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
FAX Attribute for facsimile number.	facsimileTelephoneNumber (no hyphen)	AddUser, ModifyUser, GetUserInfo	String	SYSTEM

Table 1: Attributes

Name	LDAP Mapping	Methods	Type	Directory
HOME-PHONE2 Attribute for alternative home telephone number.	omHomePhone2	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
PHONE-1 Attribute for office telephone.	telephoneNumber (no hyphen)	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
PHONE-2 Attribute for an alternative telephone.	omPhone2	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
LOCAL-UNIQUE-ID Scalix attribute identifying a unique entry in the directory. Not likely that clients will need to modify or change this.	omLocalUniqueId (no hyphen)	AddUser, ModifyUser, AddGroup, GetUserInfo	String	SYSTEM/USERLIST
GLOBAL-UNIQUE-ID Scalix attribute that globally identifies a Scalix directory entry. Not likely that client will need to modify or change this.	omGlobalUniqueId (no hyphen)	AddUser, ModifyUser, AddGroup, GetUserInfo, GetGroupInfo	String	SYSTEM/USERLIST
HOST-FQDN Scalix attribute that identifies which Scalix server owns a directory entry. Not likely that client will need to modify or change this.	omHostFqdn	AddUser, ModifyUser, AddGroup, GetUserInfo	String	SYSTEM/USERLIST
FOREIGN-ADDR Scalix attribute that identifies a foreign address. Not likely that client will need to modify or change this.	omForeignAddr (no hyphen)	AddUser, ModifyUser, AddGroup, GetUserInfo	String	SYSTEM/USERLIST
N-ID X.400 attribute for numeric identifier.		AddUser, ModifyUser	String	SYSTEM
ASSISTANT-PHONE Attribute for assistant phone number.	omAssistantPhone (no hyphen)	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
EMPL-ORG Employee organization.	omEmplOrg	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
EMPL-DEPT Employee department.	omEmplDept	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
TITLE Job title.	title	AddUser, ModifyUser, GetUserInfo	String	SYSTEM

Table 1: Attributes

Name	LDAP Mapping	Methods	Type	Directory
ADMINISTERED-BY Who administers this user account. Can have three values: scalix (SCALIX) null (not set; determined by the format of the GUID) ad (Active Directory)	omAdministeredBy (no hyphen)	AddUser, ModifyUser, GetUserInfo	String	SYSTEM
DOMAIN Domain name associated with the mailnode.		ModifyMailNode, AddMailNode, GetMailNodeInfo	String	N/A
RULE Rule associated with the mailnode.		ModifyMailNode, AddMailNode, GetMailNodeInfo	String	N/A
MAILNODE Name of the mailnode.		AddMailNode, GetMailNodeInfo	String	N/A
REFNUM Reference number by which each message ID is identified.		GetQueueMessageInfo	String	N/A
MESSAGEID Internal unique ID for messages.		GetQueueMessageInfo	String	N/A
SENDER Sender of the message in OR format.		GetQueueMessageInfo	String	N/A
RECIPIENT Recipient of the message in OR format. Can be multiple values.		GetQueueMessageInfo	String	N/A
SUBJECT Message subject.		GetQueueMessageInfo	String	N/A
SENT Date or time sent.		GetQueueMessageInfo	String	N/A
MESSAGETYPE		GetQueueMessageInfo	String {Message, Acknowledgement or Reply}	
ERRORTEXT Error text. Can be mulitple values.		GetQueueMessageInfo	String	N/A

Table 1: Attributes

Name	LDAP Mapping	Methods	Type	Directory
PRIORITY Message priority: L — Low N — Normal U — Urgent		GetQueueMessageInfo	String { "L", "N", or "U"}	N/A
ACCOUNT_STATUS Values returned from GetExtraUse-rInfo () are “locked” or “unlocked”. Value set by ModifyUser can be “lock” or “unlock”.		GetExtraUserInfo, ModifyUser	String	N/A
LAST_SIGNON Two part string:<date> <time>. This is the value last time the user LOGGEDIN in.		GetExtraUserInfo	String	Session Monitor
SERVICE_LEVEL Service level for the user. Currently not used, but will be in the future.		GetExtraUserInfo	String	N/A
SENDER Default by omaddu is true.		AddUser, ModifyUser, GetExtraUserInfo	String (true or false)	USERLIST
CAN_USE_SWA Default by ommaddu is true.		AddUser, ModifyUser, GetExtraUserInfo	String (true or false)	USERLIST
RECOVERY_FOLDER_VISIBLE Default by omaddu is false.		ModifyUser, GetExtraUserInfo	String (true or false)	USERLIST
SIS_URL Default by ommadu (URL derived where the user is provisioned).		AddUser, ModifyUser, GetExtraUserInfo	String. "" means no index auto means server derived	SYSTEM

Services and Daemons

This table shows all services and daemons available or running on the Scalix server. Use the abbreviation in the StartService and StopService interface previously outlined.

Table 2: Services and Daemons

Abbrev.	Full Name	Daemon	Service	Event Logging
ldap	LDAP Daemon	✓		
omdbmon	Database Monitor	✓		
drs	Directory Relay Server	✓		
imap	IMAP Daemon	✓		
iss	Item Structure Server	✓		
mime	MIME Browser Controller	✓		
ns	Notification Server	✓		
smtpd	SMTPD Relay	✓		
cam	Container Access Monitor	Non-stop		
idel	Item Delete Monitor	Non-stop		
lmd	License Monitor Daemon	Non-stop		
unix	Internet Mail Gateway		✓	✓
sendmail	Sendmail Interface		✓	✓
lci	Local Client Interface		✓	✓
rci	Remote Client Interface		✓	
pop	POP3 Interface		✓	✓
indexer	Search Index Server	Non-stop	✓	✓
archiver	Archive Server		✓	✓
search	Background Search Service		✓	✓
bbs	Bulletin Board Server (public folders)		✓	✓
cda	CDA Server		✓	✓
dirsync	Directory Synchronization		✓	✓
local	Local Delivery		✓	✓
omscan	Omscan Server		✓	✓

Table 2: Services and Daemons

Abbrev.	Full Name	Daemon	Service	Event Logging
print	Print Server		✓	
request	Request Server		✓	✓
router	Service Router		✓	✓
test	Test Server		✓	
admin	Administration Service		✓ (special)	✓
ubermanager	Ubermanager Service	Non-stop	✓ (special)	✓ (caa.log)
res	Remote Execution Service	Non-stop	✓ (special)	✓ (res.log)

Scalix Queues

These are the queues to be used as arguments for the queue-related interfaces.

Table 3: Scalix Queues

Queue	Description
BB	Bulletin board server input queue, meaning public folders.
DIRSYNC	Directory synchronization server input queue.
DMM	Deferred mail manager queue.
DUMP	Archive server queue.
ERRMGR	Error manager server input queue.
LICENSE	License server input queue (for virtual licenses).
LOCAL	Local delivery service input queue.
PRINT	Print server input queue.
REQ	Request server input queue.
RESOLV	Service router address resolution queue.
ROUTER	Service router input queue.
SMERR	Sendmail interface error queue.
SMINTFC	Sendmail interface input queue.
TEST	Test server input queue.
UNIX	Internet mail gateway input queue.
IDEL	The item delete queue; any message on this queue is deleted.

Table 3: Scalix Queues

Queue	Description
POISON	This queue contains messages that cause Scalix services to terminate.

Name Generation Rule Attributes

These are the friendly names of the various server-generation rules to be used in SOAP methods.

Table 4: Name Generation Rule Attributes

Rule	Formats	Comment
general.usrl_cn_rule	1 G I.S (default) 2 S, G I 3 S, G 4 G S 5 S G	1 James T.Kirk 2 Kirk, James T 3 James, Kirk 4 James Kirk 5 Kirk James
general.usrl_authid_rule	1 G I.S (default) 2 G.I.S 3 G.S 4 G_S 5 giS 6 gS 7 gis 8 S_G 9 S.G 10 S.G.I	1 James_T_Kirk@fqdn 2 James.T.Kirk@fqdn 3 James.Kirk@fqdn 4 James_Kirk@fqdn 5 jtKirk@fqdn 6 jKirk@fqdn 7 jtk@fqdn 8 Kirk_James@fqdn 9 Kirk.James@fqdn 10 Kirk.James.T@fqdn Note: The fqdn is the fully qualified domain of the host on which the user account is to be created.
orniasys.name_part_1	1 "C" <G_I_S> 2 "C" <G.I.S> 3 "C" <G.S> 4 "C" <G_S> 5 "C" <giS> 6 "C" <gS> 7 "C" <gis> 8 "C" <S_G> 9 "C" <S.G> 10 "C" <S.G.I>	Up to five rules can be defined or added to generate various Internet addresses. The rule consists of two parts: Display Name and Name part. The Display is always the CN, represented in the rule by “C”, and the smtp-email address is part included in the angle brackets.
orniasys.domain_part_1	String	For example, “scalix.com”. The attribute for the SOAP call is domain_part.
orniasys.name_part_2	Any of the above auth-id rule formats.	The attribute for the SOAP call is name_part.

Table 4: Name Generation Rule Attributes

Rule	Formats	Comment
orniasys.domain_part_2	String	
orniasys.name_part_3	Any of the above auth-id rules rule formats.	
orniasys.domain_part_4	String	
orniasys.name_part_4	Any of the above auth-id rules rule formats.	
orniasys.domain_part_4	String	
orniasys.name_part_5	Any of the above auth-id rules rule formats.	
orniasys.domain_part_5	String	
general.inet_domain_rule	String	The Scalix system domain name for the Internet address.

Server General Settings

These are the parameters or configuration settings exposed via the Scalix Management Console.

Table 5: Server General Settings

Friendly Tag	Raw Tag	Default Value	Value Type	Assoc. Service	Server or Service Restart Required
general.arch_enable_archiving Enables the archiving of all messages that traverse the Scalix server.	ARCHIVE	FALSE	String	Archiver	Archiver
general.cda_use_changelog Set this option to optimize the rebuilding of directory access tables by the CDA server. When set to TRUE, this checks the change log and builds the indexes.	CDA_USE_CHANGE_LOG	FALSE	String	cda	cda

Table 5: Server General Settings

Friendly Tag	Raw Tag	Default Value	Value Type	Assoc. Service	Server or Service Restart Required
general.imap_log_level Activates the logging of IMAP commands and errors. The log file is specified in the value IMAP_LOGFILE. Logging levels supported [1..8]. Refer to the system-wide configuration options section in the *Scalix Administration Guide* for details.	IMAP_LOGLEVEL	0	Int	IMAP	None
general.imap_log_file Name of the log file to which IMAP events are LOGGEDIN.	IMAP_LOGFILE	~/tmp/imap.%h	String	IMAP	None
general.ld_autoreply_check	LD_AUTOREPLY_CHECK_ON	TRUE	String	ld	ld
general.ld_autoreply_expiry	LD_AUTOREPLY_EXPIRY_TIME	Undefined	TBD	ld	ld
general.ndn_notify_serious Sends non-delivery reports for serious errors to the error manager only if set to TRUE. Otherwise sends to both the error manager and the originator.	NDN_EM_SERIOUS_ONLY	FALSE	String	ld	ld
general.ct_enable_ofs Specifies whether folder synchronization is enabled on the Scalix server.	OFS_ENABLED	FALSE	String	rci	rci
general.omlimit_warn_interval If the “u” sanction is enabled, the value determines the interval during which “omlimit” related messages are sent to the user. Other values that can be assigned are: 1h40m20s (1 hour 40 minutes and 20 seconds 2d30 (2 days and 40 seconds) 6000 (6000 seconds)	OMLIMIT_MIN_WARN_INTERVAL	1d	String	rci	rci

Table 5: Server General Settings

Friendly Tag	Raw Tag	Default Value	Value Type	Assoc. Service	Server or Service Restart Required
general.sr_ignore_scan Specifies the file types of items that are not scanned for viruses. Use this option to prevent certain file types from being scanned. For example, setting the value to 1167 prevents text files from being scanned. The value can be a list of colon separated numbers which map to a list of files in /var/opt/scalix/nls/language/filetype.	SR_VS_IGNORE_ITEM_TYPES	Undefined	Int	sr	sr
general.ual_disable_bb Disables or enables public access folders. If set to TRUE, user attempts to perform operations on public folders result in insufficient access rights errors.	UAL_DISABLE_BB	FALSE	String	rci	rci
general.ual_force_trace Sets the UAL trace level on system wide bases, overriding any trace value supplied by a client or set in the user.cfg. A value of 0 switches off the trace level.	UAL_FORCE_TRACE_LEVEL	0	Int	rci	rci
general.ual_keep_wastebin Setting to TRUE stops a user's waste basket from being emptied when the user signs off with the UAL.	UAL_NO_WB_EMPTY	FALSE	String	rci	rci
general.ual_pop3_trace If set, information from the in.pop3d process is traced and placed in the ~scalix/tmp directory. Setting to DETAIL generates detailed logging.	UAL_POP3_TRACE	FALSE	String	rci	rci

Table 5: Server General Settings

Friendly Tag	Raw Tag	Default Value	Value Type	Assoc. Service	Server or Service Restart Required
general.ual_signon_alias Specifies whether the alias is used after login. If set to FALSE, it reverts to Personal Name.	UAL_SIGNON_ALIAS	Undefined	String	rci	rci
general.ual_config_alias Works in conjunction with the UAL_SIGNON_ALIAS. A value of SYS means everyone can log in using an alias. A value of USER indicates the value means that alias login entries in the user.cfg. Refer to documentation for details.	UAL_SIGNON_ALIAS_CONFIG	Undefined	String	rci	rci
general.ual_use_alias Used in conjunction with UAL_SIGON_ALIAS_CONFIG and UAL_SIGON_ALIAS_CONFIG. Refer to documentation for details.	UAL_USE_SIGNON_ALIAS	FALSE	String	rci	rci
general.cn_rule Rule to generate display name or CN. Other formats supported: 1 G I.S (default) 2 S, G I 3 S, G 4 G S 5 S G	USRL_AUTO_GEN_SGI_2_CN	G I. S	String	N/A	No

Table 5: Server General Settings

Friendly Tag	Raw Tag	Default Value	Value Type	Assoc. Service	Server or Service Restart Required
general.usrl_authid_rule Rules to generate UAL authentication ID. Other formats supported: 1 G_I_S (default) 2 G.I.S 3 G.S 4 G_S 5 giS 6 gS 7 gis 8 S_G 9 S.G 10 S.G.I	USRL_AUTO_GEN_AUTHID	G_I_S	String	N/A	No
general.inet_domain_rule The system domain name. This domain is used as part of the authentication ID as well as e-mail address generation if none is specified in the rules.	INET_AUTO_GEN_DOMAIN	FQDN	String	N/A	No

Scalix Management Console Settings

See the table.

Table 6: Scalix Management Console Settings

Name	Default Value	Type
localDomain This is a multivalue. In the file, it is stored as a comma-separated value for each domain.	Default Domain (FQDN)	String
DefaultCountry Locale for the Scalix Management Console Web client.	US	String
DepartmentName This is a multivalue. In the file, each department is stored as a comma-separated value. For example: Eng, Sales, Support	""	String
MaxListSize Maximum number of entries returned for and group lists.	100	String
DefaultCity Name of the default city.	""	String
DefaultLanguage Name of the LANG variable for which message catalogues are displayed or rendered. List of supported languages: english american german	AMERICAN	String
DefaultState Default state or province.	""	String
ModifyExternalSyncedAuthId Allows users who are provisioned (omldapsync) on Scalix but externally maintained for authid modification.	false	boolean
ModifyExternalSyncedPassword Allows users who are provisioned (omldapsync) on Scalix but externally maintained for password modification. Here, the password button in the console should be enabled.	false	boolean

Error Handling

Errors and exceptions by the Management Services APIs are logged in the Tomcat log files. In addition, a UM-XXXX code returns as part of the details XML element.

A list of Management Services error message codes is provided in the table.

Table 7: Error Codes

Error Code	Error Message
UM-1000	Missing command element in the XML content received from the Management Agent
UM-1001	<root cause of exception>: Fatal Exception: Check Management Services logs
UM-1002	Received no content from the Management Agent
UM-1003	<root cause of exception>: Exception raised. Check the CAA logs
UM-1004	Failed to obtain HTTP RES connection for the server
UM-1005	Failed to obtain Management Agent URL for server from the event table
UM-1006	Missing mailnode element or its attribute in the SOAP request
UM-1007	Failed to obtain server name for mailnode
UM-1008	Mailnode is not part of Scalix server
UM_1009	Missing user type element or attribute in the SOAP request
UM-1010	Malformed userAttributes element. It must have at least a ‘CN’ element
UM-1011	Malformed userAttributes element. It must have at least a ‘CN’ element
UM-1012	Malformed userAttributes element. It must have at least ‘G’ or ‘S’ or ‘I’ elements
UM-1013	Malformed request SOAP document. Function request parameter missing ‘id’ attribute
UM-1014	No Management Agent registered from the Scalix Servers
UM-1015	Failed to locate in LDAP ID
UM-1016	Failed to locate Mailnode/HostFQDN for ID
UM-1017	Malformed request SOAP document. Missing ‘userAttributes’ for modification/addition
UM-1018	No servers are registered to accept remote messages
UM-1019	Failed to obtain CN, mailnode for all members in the request SOAP document from the LDAP server
UM-1020	Failed to retrieve host FQDN and local ID for the group from the LDAP server.
UM-1021	xxxy@zzzzuser is not authorized to perform operation xxxxx
UM-1022	Malformed request SOAP document. Missing Credentials
UM-1023	Failed to authenticate credentials for userid = xxxx@yyyy
UM-1024	‘xxxx’ is a reserved Scalix Administrative Group. Cannot delete it or modify its name

Table 7: Error Codes

Error Code	Error Message
UM-1025	'xxxx@yyyy' user is not authorized to modify passwords for Scalix administrators or for members of the Scalix Administrator group.
UM-1026	'xxxx@yyyy' user is not authorized to alter administrative privileges of a Scalix administrator
UM-1027	'xxxx@yyyy' user is not authorized to promote a member to Scalix Administrator group
UM-1028	Failed to connect to RES on host
UM-1029	Failed to obtain mailnodes from all the registered servers. Please check the administration server logs
UM-1030	Failed to contact/connect to LDAP server <server_name>
UM-1031	User/Group created successfully but failed to obtain user information from LDAP server <server_name>
UM-1032	Service or daemon argument missing in the SOAP message
UM-1033	<service_daemon> is unsupported service or daemon in Scalix
UM-1034	<service_daemon> is non-stoppable daemon in Scalix
UM-1035	No server specified in the SOAP message for the mailnode.
UM-1036	<mailnode> mailnode already exists on server <server>.
UM-1037	No mailnode specified in the SOAP message.
UM-1038	<mailnode> mailnode has users on it. Delete all users on this mailnode first.
UM-1039	You must specify a domain or rule in the SOAP message for modifying a mailnode.
UM-1040	Insufficient attributes in the SOAP Message for creating new mailnode
UM-1041	No Scalix Queue name specified in the SOAP message
UM-1042	No message reference number specified in the SOAP message
UM-1043	<queue_name> does not exist on Scalix server <server_name>
UM-1044	No server specified for this SOAP method or operation
UM-1045	No GUID specified in the SOAP message
UM-1046	Operation may have partially succeeded. Please consult the SAC and/or RES log files. Some commands failed on the following server(s):
UM-1047	You must specify service attributes in the SOAP message
UM-1048	You must specify log filters in the SOAP message
UM-1049	<service_name> service not installed on server: <server_name>
UM-1050	Insufficient rule attributes in the SOAP message Mail Address Generation Rules operation
UM-1051	Insufficient configuration attributes in the SOAP message for General configuration operation
UM-1052	Insufficient configuration attributes in the SOAP message for Password configuration operation

Table 7: Error Codes

Error Code	Error Message
UM-1053	Insufficient attributes specified in the SOAP message for Access Group Operation
UM-1054	Failed to obtain CN, Surname, Givenname, and Mailnode from LDAP for id = <id>
UM-1055	You cannot manage Scalix Administration Groups
UM-1056	Insufficient configuration attributes in the SOAP message for Password configuration operation
UM-1057	<department_name> department does not exist
UM-1058	Failed to update Departments configuration file. Check SAC error log file.
UM-1059	Insufficient attributes specified in the SOAP message for departmental operation
UM-1060	<department_name> department already exists. Please choose another name
UM-1061	<user_name> is a non Scalix user. Cannot fetch extra information for such users
UM-1062	Failed to obtain or locate registered server table entry for server
UM-1063	Failed to modify ubermanager.properties file. Check SAC error log file.
UM-1064	Insufficient arguments in the SOAP message for User Conversion for GUID <guid>
UM-1065	User Conversion for type of user specified in the SOAP message is not supported yet.
UM-1066	User Conversion: Failed to obtain save entry for guid:host <guid:host>
UM-1067	User Conversion: Failed to obtain cache entry for guid:host <guid:host>
UM-1068	User Conversion: Failed to update cache entry for guid:host <guid:host>
UM-1069	User Conversion: Failed to make master entry for guid:host <guid:host>
UM-1070	User Conversion: Failed to obtain remote server for mailnode <guid:host>
UM-1071	Missing 'type' attribute for SOAP message
UM-1072	<type> type Scalix License is not supported
UM-1073	Insufficient attributes specified in the SOAP message for the License Add operation
UM-1074	<mailnode> mailnode already exists on registered Scalix server <server>
UM-1075	<server> RES may not have equivalent functionality implemented. Please upgrade to <version>
UM-1076	Malformed SOAP message for Resource operation. Missing ‘resourceAttributes’ element or none specified
UM-1077	Malformed SOAP message for Resource operation. Must have at least CN or Surname
UM-1078	Insufficient attributes specified in the SOAP message for Resource Add/Modify operation
UM-1079	Missing authid attribute for this SOAP message call
UM-1080	Missing Plugin 'name' attribute for this SOAP message call
SX 1	Illegal usage of command

Table 7: Error Codes

Error Code	Error Message
SX 2	Python Exception: Check scalix-res.log file
SX 3	Invalid License. The license either has been tampered with or is not valid