STRONGVIEW

MESSAGE STUDIO 8.1
SOAP API Reference Guide

PUBLISHED BY:

US Headquarters

1300 Island Drive, Suite 200
Redwood City, CA 94065
United States
P: +1 (650) 421-4200
F: +1 (650) 421-4201

UK Headquarters

StrongView Systems UK, Ltd
Adelaide House
Perth Industrial Estate
Slough
Berkshire SL1 4XX
United Kingdom
P: +44 (0) 203 131 0144

APAC Headquarters

XCOM Media
Unit 1
15 Lamington St.
New Farm
Queensland 40005
Australia
P: +61 7 3666 0544

Copyright © 2014 StrongView Systems, Inc. All rights reserved

No part of the contents of this publication may be reproduced or transmitted in any form or by any means without the written permission of StrongView Systems, Inc.

The information furnished herein is believed to be accurate and reliable. However, no responsibility is assumed by StrongView Systems for its use, or for any infringements of patents or other rights of third parties resulting from its use.

STRONGVIEW and the STRONGVIEW logo are registered trademarks in the United States, other countries or both. All Rights Reserved.

StrongView Systems UK, Ltd is a company registered in England and Wales at 5 New Street Square, London EC4A 3TW. Reg. No. 6398867. VAT # GB 925 07 6228. Trading Address: Adelaide House, Perth Industrial Estate, Slough, Berkshire, SL1 4XX.

http://www.strongview.com

Tuesday, January 06, 2015

Contents

Message Studio SOAP API Introduction

The Message Studio Web Services Application Programming Interface (WS API) provides access to the data, content, and campaign management assets and features that are available in a Message Studio instance, allowing developers to build custom messaging applications and closely integrated business systems.

Topics:

Additional Resources

- **Developer Central:**

  http://spark.strongview.com/community/developers?view=overview
- **Code Sharing:**

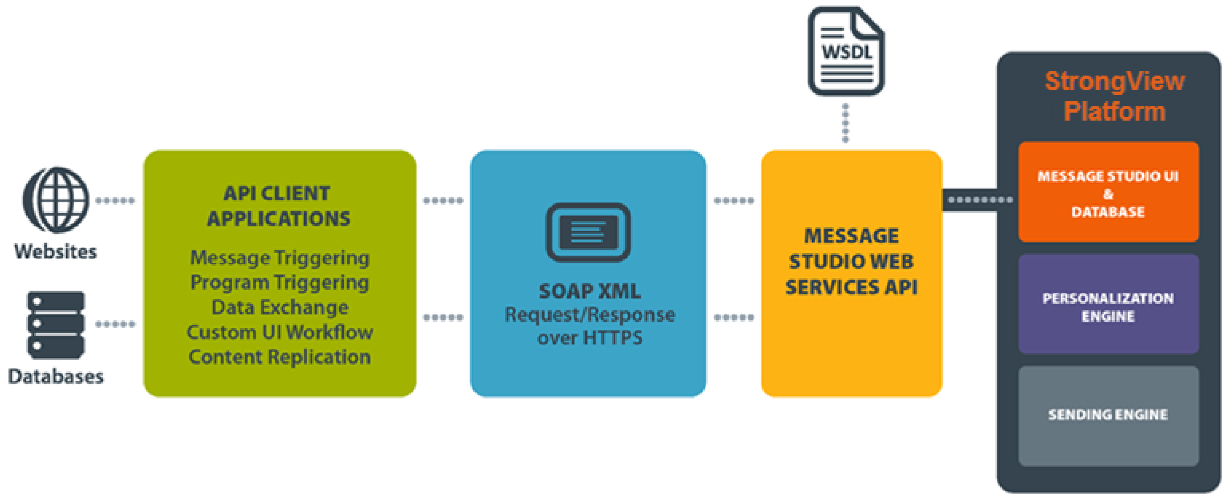
  http://spark.strongview.com/community/developers/code_sharing
- **API Q&A:**

  http://spark.strongview.com/community/developers/q%26a

WS API Supported Functionality

The Message Studio Web Services Application Programming Interface (WS API) provides access to the data, content, and campaign management assets and features that are available in a Message Studio instance, allowing developers to build custom messaging applications and closely integrated business systems.

Figure 1: Web Services API Concept



The WS API supports a subset of the functionality of the Message Studio user interface as described in the following table.

Functional Area	Supported Message Studio Function
Triggering Transactional Mailings and Lifecycle Marketing Messages	• Adding participants to a program • Triggering a transactional message send • Loading, pausing, and resuming transactional mailings
Data Management	• Creating, copying, and deleting data sources, seed lists, suppression lists, and targets
Content Management	• Creating, copying. and deleting message templates • Testing message templates • Validating XSL-based templates • Managing message attachments, content blocks, and dynamic content rules
Campaign Management	• Creating, copying. and deleting batch and transactional mailings • Launching test mailings • Launching, pausing, and resuming mailings • Retrieving mailing status and performance metrics • Closing and archiving mailings

Functional Area	Supported Message Studio Function
User, Organization, and System Management	• Retrieving and listing users, roles, and organizations in Message Studio instance • Retrieving system information such as bounce, from, reply-to addresses, campaign categorization, and media servers

When to Use the Web Services API

StrongView provides three distinct API sets for its product stack:

- Message Studio Web Services API
- Email Integration Server (EIS) Batch API
- Email Integration Server (EIS) Transactional API

In general, you should build your API client applications using the Message Studio API since it offers a modern standards-based Web services programming interface and the broadest set of capabilities for Message Studio customization and automation. You should avoid using both the Message Studio and EIS APIs together. For example, assets created with EIS APIs will not be available via the Message Studio UI or API.

In some cases, you may need to use the EIS-based APIs:

- **High Performance Transactional Messaging**—If you need to support a very large volume of transactional messages (greater than 500,000 messages per hour), then you may need to use the EIS Transactional API. Please contact StrongView Solutions Consulting Group to validate the appropriate API for this use case.
- **EIS/EDS-Only Customers**—If you are not using the full Message Studio stack, then you can use the EIS APIs to create and manage mailings. Please refer to the *StrongView EIS API Reference Guide* for more information.

Platform Maintenance and Downtime

The StrongView servers will usually need to have maintenance downtimes on a recurring schedule. During these downtimes, the Message Studio API is not available. Attempts to create an API session will return a system downtime exception and client applications will need to take the appropriate action, which may include alerts to support staff, integration job queuing, and/or scheduled re-attempts.

System Requirements

Please make sure you are familiar with the system requirements before beginning.

Hardware Requirements

Message Studio API's are lightweight and do not require significant processing power or memory. However, a high-speed Internet connection is recommended for the services to be most effective.

Knowledge Requirements

- Some familiarity with SOAP and web services architecture.
- Ability to program in some language.
- Familiarity with XML may be helpful for troubleshooting.

Backward Compatibility and Versioning

The system supports backward compatibility as new versions of the StrongView Web Services API are released. The WS API is backward compatible in that an application created to work with a given WS API version will continue to work with that same WS API version in future platform releases.

Specifically, the WS API endpoint URL is versioned so that different WS API versions have different endpoint URLs. Your applications will continue to work with the WS API endpoint URLs of previous releases and you have the opportunity to migrate your client applications to the newer WS API version endpoint URLs to leverage enhanced functionality on a schedule that meets your needs.

StrongView does not guarantee that applications written against one WS API version will work with future API versions, since changes in method signatures and data representations are often required as we continue to enhance the StrongView platform. However, we strive to keep the WS API consistent from version to version with minimal if any changes required to port applications to newer WS API versions. When an API version is to be deprecated, advance end-of-life notice will be given at least 12 months before support for the API version ends.

StrongView currently supports the following versions of the WS API:

WS API Version	Endpoint	End-of-Life Date
1	/sm/services/mailing/2009/03/02	2017-07
2	/sm/services/mailing/v2	None announced

Standards Compliance

The StrongView WS API is in compliance with the following specifications.

Standards Name	Website
Simple Object Access Protocol (SOAP) 1.1	**http://www.w3.org/TR/2000/NOTE-SOAP-20000508/**
Web Service Description Language (WSDL) 1.1	**http://www.w3.org/TR/2001/NOTE-wsdl-20010315**
WS-I Basic Profile 1.1	**http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html**

Security Considerations

Refer to the following topics with regards to web service security.

Securing API Communications with SSL

Communications between your client application and the StrongView WS API can be secured by using the Secure Sockets Layer (SSL) protocol for HTTP requests and responses. This is a standard approach for securing internet-based communications and was designed to prevent eavesdropping, tampering, and message forgery. It secures communications over the internet by cryptographic methods based on a 128-bit encryption and is a standard approach for securing SOAP-based Web service transactions.

User Authentication

If you already have a Message Studio account, you can use the credentials from that account to access web services. Your access permissions to the application through web services mirrors your access from the UI.

For instance, if you don’t have the ability to create mailings within a particular organization in the UI, you will not be able to create mailings within that organization using web services. For more information about users and access privileges, please read the **Message Studio User Guide**.

Note: Remember that the organization in the SOAP header indicates the organization where you are performing the operation. For example, if you want to create a mailing in the admin organization, then you must log into (through the SOAP header) the admin organization. If you then want to create a mailing in the IDB organization, you will log into (through the SOAP header) the IDB organization.

User Credentials

To access Message Studio via API, you will need the same account information as you need to log into the UI. Make sure you have the Organization name, User name, and password. The Organization provides the context for the action you're performing through the API - whatever action you are taking is based on the user logging in.

Note: In most cases, StrongView recommends you make API calls as a super-user, which will provide access to the entire system.

All communication is similar to the login process for the UI.

SOAP uses the organization, login name, and password provided in this security header to authenticate a user to the application.

Security Header

Access to Message Studio's API is granted based on the contents of the security header, which is required for every call into the server. All communication is over https, which provides encryption and requires a self-signed SSL certificate.

Message Studio API is stateless. This means that every SOAP requests into Message Studio must include a security header that provides the user credentials. Every request is treated as follows:

1. Login.
2. Perform single action.
3. Logout (explicit).

Sample SOAP Header

The following is an example of the security tags you need to place in the header of each SOAP request. You will want to replace the username, password and organization with your personal login information.

The sub-organization ID is optional and identifies the organization or sub-organization you are accessing.

```
/*
* ------------------------------------------------------------------ *
*  STRONGVIEW SYSTEMS                                                *
*                                                                    *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                    *
*  Visit http://www.strongview.com for more information             *
*                                                                    *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                    *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                            *
*                                                                    *
* ------------------------------------------------------------------ *
*/
<soap:Header>
  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
    <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/ oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-9243153">
      <wsse:Username xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis- 200401-wss-wssecurity-secext-1.0.xsd">admin</wsse:Username>
      <wsse:Password xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis- 200401-wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">mypassword</wsse:Password>
    </wsse:UsernameToken>
    <OrganizationToken xmlns=”http://www.strongmail.com/services/2009/03/02/schema”>
      <organizationName>IDB</organizationName>
      <subOrganizationId>
        <id>2601</id>
      </subOrganizationId>
    </OrganizationToken>
  </wsse:Security>
```

```
</soap:Header>
```

Firewall

You can limit Web service access to the APIs by modifying your iptables configuration to only allow certain IPs to access the Web services port. To do this:

1. Edit the iptable configuration file:

   ```
   vi /etc/sysconfig/iptables
   ```

2. Before the REJECT line, add ACCEPT lines to allow access. For example, the following lines allow users from IP 192.168.21.165 to connect the Web services port (4443) and rejects all other IPs:

   ```
   iptables -A RH-Firewall-1-INPUT -s 192.168.21.165 -p tcp -- dport 4443 -j
   ACCEPT
   -A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
   ```

3. Restart the iptables service:

   ```
   service iptables restart
   ```

IPtables Example

Here is an example of a complete iptables configuration, with a single IP (192.168.21.165) allowed access to the Web services port.

```
# Generated by iptables-save v1.3.5 on Thu Feb 19 20:40:26 2009 *filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [10240089:5452482095] :RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp -m icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p esp -j ACCEPT
-A RH-Firewall-1-INPUT -p ah -j ACCEPT
-A RH-Firewall-1-INPUT -d 227.0.0.152 -p udp -m udp --dport 5353 -j ACCEPT -A
RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A RH-Firewall-1-INPUT -s 192.168.21.35 -p tcp -m tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -s 10.0.0.0/23 -p tcp -m tcp --dport 22 -j ACCEPT
```

```
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 25 -j
ACCEPT -A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 80 -
j ACCEPT -A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport
4443 -j ACCEPT -A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --
dport 9000 -j ACCEPT -A RH-Firewall-1-INPUT -s 192.168.21.87 -p udp -m udp --
dport 161 -j ACCEPT
-A RH-Firewall-1-INPUT -s 10.0.3.0/25 -p udp -m udp --dport 161 -j ACCEPT
-A RH-Firewall-1-INPUT -s 10.0.185.0/27 -p udp -m udp --dport 161 -j ACCEPT
-A RH-Firewall-1-INPUT -s 192.168.21.165 -p tcp -m tcp --dport 9443 -j ACCEPT
-A RH-Firewall-1-INPUT -s 192.168.21.187 -p tcp -m tcp --dport 4443 -j ACCEPT
-A RH-Firewall-1-INPUT -s 192.168.21.84 -p tcp -m tcp --dport 5432 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Thu Feb 19 20:40:26 2009
```

Getting Started

The following procedure steps you through interfacing web service applications with Message Studio through the API.

See Also:

- **Development Environments on page 18**
- **Sample Java Code on page 34**
- **Sample C# Code on page 46**
- **XML Examples on page 54**
- **Frequently Asked Questions on page 75**
- **Reference on page 74**

Step 1: Obtain the WSDL

First, retrieve and inspect the WSDL for the Message Studio API so that you are familiar with the interface and prepared to start development using the various programming language-specific SOAP libraries.

The WSDL can be found at the following location on the Internet:

```
https://<hostname>/sm/services/mailing/version?wsdl
```

where:

version—Version of the API. Example: `2009/03/02,v2`

Step 2: Obtain the Sample Code

Next, you can get a head start on your project by retrieving the sample code bundle and using the common code snippets and patterns that are provided in the bundle.

Sample code is available on the Message Studio server in the following directory:

```
/data1/message_studio/lib/sm/webservice_samples.tar.gz
```

In addition, you can access the code from a browser:

```
https://hostname/sm/webservice_samples.tar.gz
```

where:

hostname—Hostname of your Message Studio server

Note: In order to view the sample code, you must be logged into Message Studio.

Step 3: Obtain a SOAP Client Library for your Development Environment

Use of a SOAP Client Library allows you to more easily generate and process the SOAP XML requests and responses that form the basis of communication with the Message Studio API. StrongView has tested the following SOAP Client Libraries with its sample code.

Language	Recommended SOAP Client
Java	CXF, Axis2
C#	.NET 4.5
PHP	wsdlInterpreter
Ruby	SOAP4r
Python	Suds

You should use these SOAP Client Libraries to generate the supporting code for your application. For details on how to use or generate the supporting code, refer to the documentation provided for these libraries or browse the StrongView Spark community site for tips on how to best use these libraries.

Step 4: Walk-through a Simple Example

The general pattern of a Java-based client application for the Message Studio Web Services is provided below:

- Create a web service object that you will use to make calls in your client application. Creating this service object involves establishing WSDL and XSD references and the organization, user, and password which will be added as part of the SOAP security header to authenticate each API call to the Message Studio Web Service.

- Create a request object and set member values to accomplish a given task.

- Make the desired API call on the service object and pass the request object.
- Parse the response object to extract the desired information or confirm success of an API call.
- Catch any errors that are thrown.
- Continue to create request objects, make calls on service objects, and parse response objects to accomplish the goals of the API client application.

Development Environments

StrongView recommends Java CXF for the Message Studio web services. This section provides information about setting up your CXF development environment.

Topics:

Java CXF Clients

By generating your Java stubs with CXF, you can quickly build and develop Message Studio services.

Topics:

Generating Java Client Stubs

The WSDL can be directly translated to a set of Java classes that provide helper classes to hold parameter values, serialize the values as XML, and make the necessary HTTP requests to the SOAP server. This is done using the `wsdl2java` tool, which is part of the Apache CXF open source web services framework.

You can also use optional arguments to the `wsdl2java` tool to customize the generated code, as well as to generate an Ant based makefile to build your application.

Note: The wsdl2java options and output referenced here assume use of Apache CXF 2.1.3. If you are using a different version of CXF or a different web services framework altogether, the command options and results may be different.

StrongView recommends running the following command to generate the Java code:

```
wsdl2java -b bindingClient.xjb -d . -p
http://www.strongmail.com/services/2009/03/92=com.strongmail.
webserviceclient.samples.cxf.generated -p
http://www.strongmail.com/services/2009/03/02/schema=
com.strongmail.webserviceclient.samples.cxf.generated.objects -p
http://www.w3.org/2001/XMLSchema=
com.strongmail.webserviceclient.samples.cxf.generated.objects.xmlschema -sn
MailingService -validate -wv 1.1 MailingService.wsdl
```

These options provide you the following:

- `-b` *`binding`*—This option specifies an external binding file. `bindingClient.xjb` is available in the `webservices_samples/cxf` directory.
- `-d` *`output_directory`*—This option specifies the directory into which the Java code is written. If this is not explicitly specified, the files will be placed in the directory from which the `wsdl2java` command is invoked.
- `-p` *`path`*—This option specifies a subdirectory to place generated files for the named package.
- `-sn` *`webservice`*—This option specifies the name of the web service.
- `-validate`—This option validates the WSDL before generating the code.
- `-wv` *`version`*—This option specifies the version of the WSDL.

The generated code directory structure is:

Directory	Description
com/strongmail/webserviceclient/samples/cxf/generated	This directory contains a the generated MailingService interface and the MailingService_Service implementation. The MailingService interface defines all methods you can call. It also contains Java classes representing the fault definitions.
com/strongmail/webserviceclient/samples/cxf/generated/objects	This directory contains a Java class implementation for each type defined in the XSD.

Constructing a Java SOAP Client

Once you have generated the client stubs using wsdl2java, you can begin constructing a SOAP client. The sample code creates a service in two steps, which makes it easy to set different headers for different organizations in subsequent calls:

1. Create a raw service for a given endpoint. You can specify an endpoint or use the one in the WSDL. At this point, the created service won’t add headers to

operations.

2. Apply the authentication interceptor to the service to set username, password, and organization (and optionally, the sub-organization). After these two steps, a service will automatically add the same header to each operation. If you want to change the header to log into a different organization, you just need to re-apply the interceptor with new login information.

See these methods for the code:

- `ClientSetup.java: public MailingService createService(File wsdlFile, String endpointAddress) throws Exception;`
- `SampleProgram.java: public static MailingService createService();`
- `ClientSetup.java: public void applyInterceptor(MailingService svc, String username, String password, String organizationName, Long subOrganizationId) throws Exception;`

Java Security Token Requirements

A security token header must be passed with every API call. This header needs to contain username, password and organization (which is the same information needed in order to login to the Message Studio UI).

In CXF, you set security by using an interceptor. The interceptor will be implicitly called when you call each operation. The benefit of this approach is that you only need to set login information once, instead of passing it to each operation call.

Since the security header is defined by WS-Security specification and is handled automatically by the CXF library, you only need to pass in the properties to the interceptor.

The following code is required to pass Message Studio API authentication which uses WS-Security SOAP headers. You can use the following code to set up the interceptor.

```
/*
* -------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information               *
*                                                                      *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                      *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                              *
*                                                                      *
* -------------------------------------------------------------------- *
*/

/* WSSE standard username token */
Map<String, Object> outprops = new HashMap<String, Object>();
outprops.put(WSHandlerConstants.ACTION, WSHandlerConstants.USERNAME_TOKEN);

/* Set username and password */
outprops.put(WSHandlerConstants.USER, username);
outprops.put(WSHandlerConstants.PASSWORD_TYPE, WSConstants.PW_TEXT);
outprops.put(WSHandlerConstants.PW_CALLBACK_REF, newClientPasswordCallbackHandler
(password));

/* Create client interceptor AuthenticationInterceptor */
authenticationInterceptor = new AuthenticationInterceptor(schemaNS, outprops,
organizationName, null);
```

The following code is included in the sample code (`AuthenticationInterceptor.java`) and extends the CXF interceptor. It sets the `UsernameToken` (username and password) by calling the parent CXF interceptor. Since the `OrganizationToken` (`organizationName/ subOrganizationId`) is not standard, but defined by StrongView, you need to handle them manually. You can do this by defining an inner class and adding it to the interceptor chains.

```
/*
* -------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information                *
*                                                                      *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                      *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                              *
*                                                                      *
* -------------------------------------------------------------------- *
*/
```

```
package com.strongmail.webserviceclient.samples.cxf;
import org.apache.cxf.binding.soap.SoapMessage;
import org.apache.cxf.interceptor.Fault;
import org.apache.cxf.phase.Phase;
import org.apache.cxf.phase.PhaseInterceptor;
import org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor;
import org.apache.ws.security.SOAPConstants;
import org.apache.ws.security.handler.WSHandlerConstants;
import org.apache.ws.security.util.WSSecurityUtil;
import org.w3c.dom.Element;
import org.w3c.dom.Text;
import java.util.Collections;
import java.util.Map;
import java.util.Set;
import javax.xml.soap.SOAPException;
import javax.xml.soap.SOAPMessage;
public class AuthenticationInterceptor extends WSS4JOutInterceptor {

  /** Schema name space */
  private String schemaNS;

  /** Interceptor to set headers: organizationName and subOrganizationId */
  private PhaseInterceptor phaseInterceptor;

  /** organizationName and subOrganizationId to login */
  private String organizationName;
  private Long subOrganizationId;
  public AuthenticationInterceptor(String schemaNS, Map<String, Object> properties,
String orgName, Long subOrganizationId)
  {
    super(properties);
    this.schemaNS = schemaNS;
    this.phaseInterceptor = new WSS4JOutInterceptorInternal();
    this.organizationName = orgName;
    this.subOrganizationId = subOrganizationId;
  }

  public void setOrganizationName(String orgName) {
    this.organizationName = orgName;
  }

  public void setSubOrganizationId(Long subOrganizationId) {
    this.subOrganizationId = subOrganizationId;
  }
  public void setUsername(String username) {
    getProperties().put(WSHandlerConstants.USER, username);
  }

  public void setPassword(String password) {
    getProperties().put(WSHandlerConstants.PW_CALLBACK_REF, new
ClientPasswordCallbackHandler(password));
  }

  public void handleMessage(SoapMessage message) throws Fault {
    super.handleMessage(message);
    message.getInterceptorChain().add(phaseInterceptor);
  }

  /**
  * Set SOAP header before sending out message.
  * Header includs username, password, organizationName and optionally subOrganizationId.
  * This message set organizationName and subOrganizationId, and parent class will set
username/password.
  */

  class WSS4JOutInterceptorInternal implements PhaseInterceptor<SoapMessage> {
    private static final String ORGANIZATION_TOKEN_ELEMENT_NAME = "OrganizationToken";
    private static final String ORGANIZATION_ELEMENT_NAME = "organizationName";
    private static final String SUB_ORGANIZATION_ID_ELEMENT_NAME = "subOrganizationId";
    private static final String ID_ELEMENT_NAME = "id";

    public WSS4JOutInterceptorInternal() {
      super();
    }

    public void handleMessage(SoapMessage message) {
      try {
        Element securityHeader = getSecurityHeader(message);

        /* Create <OrganizationToken> and <organizationName> element, */
        /* and set <organizationName> as child of <OrganizationToken> */
        Element e = securityHeader.getOwnerDocument().createElementNS(schemaNS,
ORGANIZATION_TOKEN_ELEMENT_NAME);
        Element e2 = securityHeader.getOwnerDocument().createElementNS(schemaNS,
ORGANIZATION_ELEMENT_NAME);
        Text t = securityHeader.getOwnerDocument().createTextNode(organizationName);
        e2.appendChild(t);
        e.appendChild(e2);
        /* Create <subOrganizationId> if necessary. It is also child of
<OrganizationToken> if (subOrganizationId != null) /*
        {
          Element e3 = securityHeader.getOwnerDocument().createElementNS( schemaNS, SUB_
ORGANIZATION_ID_ELEMENT_NAME);
          Element e4 = securityHeader.getOwnerDocument().createElementNS(schemaNS, ID_
ELEMENT_NAME);
          Text t2 = securityHeader.getOwnerDocument().createTextNode(
subOrganizationId.toString());
          e4.appendChild(t2);
          e3.appendChild(e4);
          e.appendChild(e3);
        }
        securityHeader.appendChild(e);
      }
      catch (Throwable e)
      {
        e.printStackTrace();
      }
    }
    private Element getSecurityHeader(SoapMessage message) throws SOAPException {
      SOAPMessage doc = message.getContent(SOAPMessage.class);
      String actor = (String)getOption(WSHandlerConstants.ACTOR);
      SOAPConstants sc = WSSecurityUtil.getSOAPConstants(doc.getSOAPPart
().getDocumentElement());
      return WSSecurityUtil.getSecurityHeader(doc.getSOAPPart(), actor, sc);
    }
    public Set<String> getAfter()
    {
      return Collections.emptySet();
    }
    public Set<String> getBefore()
    {
      return Collections.emptySet();
    }
    public String getId()
    {
      return WSS4JOutInterceptorInternal.class.getName();
    }
    public String getPhase()
    {
      return Phase.POST_PROTOCOL;
    }
    public void handleFault(SoapMessage message) {
    /*nothing */
    }
    }
}
```

C# Clients

C# is a multi-paradigm programming language that has an object-oriented syntax.

Topics:

- **Generating C# Client Stubs on page 25**
- **Creating a C# Mailing Service on page 25**
- **Constructing a C# SOAP Client on page 28**
- **C# Security Token Requirements on page 30**

Generating C# Client Stubs

Stubs for a C# implementations may be generated from the provided WSDL/XSD files, as follows:

```
wsdl /l:CS /protocol:SOAP MailingService.wsdl MailingServiceSchema.xsd
```

Once the stubs have been generated, a change must be made to one of the generated files. In the generated file `MailingService.cs`, you must update the MailingService class definition so that it extends the class:

```
Microsoft.Web.Services2.WebServicesClientProtocol
```

instead of the class:

```
System.Web.Services.Protocols.SoapHttpClientProtocol
```

Creating a C# Mailing Service

The following function can be used to create and return a service:

```
/*
* -------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information                *
*                                                                       *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                       *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                               *
*                                                                       *
* -------------------------------------------------------------------- *
*/

public const string url = "https://192.168.29.160:4443/sm/services/mailing/2009/03/02";
private static MailingService createService() {
  /* Create service and set endpoint */
  MailingService svc = new MailingService(); svc.Url = url;

  /* Ignore Certificate by trusting all certificate */
  System.Net.ServicePointManager.CertificatePolicy = new TrustAllCertificatePolicy();

  /* Create security tokens for SOAP header */
  UsernameToken userToken = new UsernameToken("admin@strongview.com", "admin",
PasswordOption.SendPlainText);
  SoapContext requestContext = svc.RequestSoapContext;
  requestContext.Security.Tokens.Add(userToken);

  /* Use filter to set organization token and get rid of timestamp since it does not work
with Java server. */
  MailingServiceOutFilter myfilter = new MailingServiceOutFilter("admin", ns);
  svc.Pipeline.OutputFilters.Insert(0, myfilter);
  return svc;
}
```

After creating the mailing service, you can call operations on it. For example:

```
MailingService svc = createService()
```

Security token requirements A security token header must be passed with every API call. This header needs to contain username, password and organization (which is the same information needed in order to login to the Message Studio UI).

The default Microsoft implementation of a security header token already supports username and password, so once you have created a `MailingService` instance, it is straight-forward to add username and password to its header:

```
MailingService svc = new MailingService();
```

```
    UsernameToken userToken = new UsernameToken("admin@strongview.com", "admin",
    PasswordOption.SendPlainText);
    SoapContext requestContext = svc.RequestSoapContext;
    requestContext.Security.Tokens.Add(userToken);
```

However, you must also add an organization to the header, which must be done manually. In addition, by default Microsoft adds a timestamp to the header which is not handled correctly by the Java server-side code and which will cause your API calls to fail if it is not removed.

Therefore, you must remove the Microsoft timestamp manually. You can take care of both of these items by defining a custom filter and adding the desired code within its `ProcessMessage()` method. For example:

```
    MailingServiceOutFilter myfilter = new MailingServiceOutFilter("myOrg");
    svc.Pipeline.OutputFilters.Insert(0, myfilter);
```

Then, the `ProcessMessage()` method might be:

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information            *
*                                                                  *
*  You may incorporate this Source Code in your application only   *
*  you own a valid license to do so from StrongView Systems, Inc   *
*  and the copyright notices are not removed from the source cod   *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/
/* Set internal namespace field */
_ns = http://www.strongmail.com/services/2009/03/02/schema;
public override void ProcessMessage(SoapEnvelope envelope) {

  /* Add OrganizationName token */
  XmlElement OrgTokenElement = envelope.CreateElement("sm", "OrganizationToken", _ns);
  XmlElement orgElement = envelope.CreateElement("sm", "organizationName", _ns);

  /* Set organizationName value */
  XmlText orgName = envelope.CreateTextNode(_organization);
  orgElement.AppendChild(orgName);
  OrgTokenElement.AppendChild(orgElement);

  /* Remove timestamp */
  IEnumerator it = envelope.Header.GetEnumerator();
  while(it.MoveNext()){
    XmlElement el = (XmlElement) it.Current;
    if (el.Name.Equals("wsse:Security")) {
      el.AppendChild(OrgTokenElement);

      /* Remove Timestamp */
      XmlNodeList timestamp = el.GetElementsByTagName("wsu:Timestamp");
      el.RemoveChild(timestamp.Item(0));
    }
  }
}
```

Constructing a C# SOAP Client

To create the SOAP client, you will need the following script:

```
/*
* -------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information                *
*                                                                      *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                      *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                              *
*                                                                      *
* -------------------------------------------------------------------- *
*/

public static MailingServiceStub createService() {

  /* Read key store, key store password, wsdl file location and endpoint address */
  /* Endpoint address will replace the one in WSDL file  */
  System.out.println("Store: " + System.getProperty("javax.net.ssl.trustStore"));
  System.out.println("Password: " + System.getProperty
("javax.net.ssl.trustStorePassword"));
  System.out.println("endpoiont: " + System.getProperty("endpoint.address"));

  /* File */
  wsdlFile = new File(System.getProperty("wsdl"));
  String endpoint = System.getProperty("endpoint.address");
  System.getProperty("services.namespace");
  String schemaNs = System.getProperty("services.namespace.schema");
  try {

    /* Create service */
    MailingServiceStub stub = new MailingServiceStub(endpoint);

    /* Add security header */
    /* username/password/organization are the same as UI login. */
    OMFactory fac = OMAbstractFactory.getOMFactory();
    OMNamespace omNs = fac.createOMNamespace(WSSE_NAMESPACE,"wsse");
    OMNamespace smSchemaNS= fac.createOMNamespace(schemaNs, "smschema");

    /* Create Security and UsernameToken */
    OMElement security = fac.createOMElement("Security", omNs);
    OMElement token = fac.createOMElement("UsernameToken", omNs);

    /* Set username */
    OMElement username = fac.createOMElement("Username", omNs);
    OMText usernametext = fac.createOMText("admin@strongview.com");
    username.addChild(usernametext);

    /* Set password */
    OMElement password = fac.createOMElement("Password", omNs);
    password.addAttribute("Type", PASSWORD_TYPE, null);
    OMText passtext = fac.createOMText("admin");
    password.addChild(passtext);
    token.addChild(username);
    token.addChild(password);
    security.addChild(token);

    /* Add OrganizationToken */
    OMElement orgtoken = fac.createOMElement("OrganizationToken", smSchemaNS);
    OMElement orgname = fac.createOMElement("organizationName", smSchemaNS);
    OMText orgtext = fac.createOMText("admin");
    orgname.addChild(orgtext);
    orgtoken.addChild(orgname);
    security.addChild(orgtoken);
    stub._getServiceClient().addHeader(security); return stub;
  }
  catch (Exception e) {
    e.printStackTrace();
    System.exit(1);
  }
  return null;
}
```

C# Security Token Requirements

A security token header must be passed with every API call. This header needs to contain username, password and organization (which is the same information needed in order to login to the Message Studio UI).

The following code is required to pass Message Studio API authentication which uses WS-Security SOAP headers. This code is included in the sample code, and is available in `Sample.Program.java`; refer to the `createService` method.

```
/*
* -------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information                *
*                                                                      *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                      *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                              *
*                                                                      *
* -------------------------------------------------------------------- *
*/

/* Create service */
MailingServiceStub stub = new MailingServiceStub(endpoint);

/* Add security header */
/* username/password/organization are the same as UI login. */
OMFactory fac = OMAbstractFactory.getOMFactory();
OMNamespace omNs = fac.createOMNamespace(WSSE_NAMESPACE, "wsse");
OMNamespace smSchemaNS = fac.createOMNamespace(schemaNs, "smschema");

/* Create Security and UsernameToken */
OMElement security = fac.createOMElement("Security", omNs);
OMElement token = fac.createOMElement("UsernameToken", omNs);

/* Set username */
OMElement username = fac.createOMElement("Username", omNs);
OMText usernametext = fac.createOMText("admin");
username.addChild(usernametext);

/* Set password */
OMElement password = fac.createOMElement("Password", omNs);
password.addAttribute("Type", PASSWORD_TYPE, null);
OMText passtext = fac.createOMText("admin");
password.addChild(passtext);
token.addChild(username);
token.addChild(password);
security.addChild(token);

/* Add OrganizationToken */
OMElement orgtoken = fac.createOMElement("OrganizationToken", smSchemaNS);
OMElement orgname = fac.createOMElement("organizationName", smSchemaNS);
OMText orgtext = fac.createOMText("admin");
orgname.addChild(orgtext);
orgtoken.addChild(orgname);

/* If you need to log in to a suborganization, then uncomment */
/* the following block, and change the string "100" to be a */
/* string containing the id of the suborganization you want. */
OMElement subOrgId = fac.createOMElement("subOrganizationId", smSchemaNS); OMElement id =
fac.createOMElement("id", smSchemaNS);
OMText subOrgIdText = fac.createOMText("100");
id.addChild(subOrgIdText);
subOrgId.addChild(id);
orgtoken.addChild(subOrgId);
security.addChild(orgtoken); stub._getServiceClient().addHeader(security);
```

Ruby Clients

Ruby is a general purpose object-oriented programming language that is often used to develop web services.

Note: The information provided in this section assumes that you will be developing Ruby code in a Windows environment.

In preparing your development environment, perform the following steps:

1. On your Windows machine, install the following:

Installer	Download From
`ruby186-25.exe`	**http://rubyforge.org/frs/download.php/76955/rubyinstaller-2.0.0-p195.exe**
`ruby-gnome2-0.16.0-1-i386-mswin32.exe`	**http://rubyforge.org/frs/download.php/76955/rubyinstaller-2.0.0-p195.exe**
soap4r	**https://rubygems.org/downloads/mumboe-soap4r-1.5.8.7.gem**
httpclient	**https://rubygems.org/downloads/httpclient-2.3.4.1.gem**

2. After installing the clients, run the following commands:

```
ruby_install_path]/bin/gem install httpclient-2.3.4.1.gem

ruby_install_path]/bin/gem install mumboe-soap4r-1.5.8.7.gem
```

Generating Ruby Client Stubs

Create the SOAP wrappers from Message Studio WSDL. This should create all of the required client ruby classes used to communicate with StrongView SOAP server.

1. `$ cd <path>\lib`
2. `$ ruby <soap4r1.5.8 path>/wsdl2ruby.rb --type client --wsdl <url to wsdl>`

Constructing a Ruby SOAP Client

Use the following code to construct the SOAP client.

```
def self.sm_service(opts={})
# create the service
```

```
      obj = MailingService.new(opts[:endpoint_url])
      # run ruby with -d to see SOAP wiredumps.
      obj.wiredump_dev = STDERR if $DEBUG
      # attach auth header handler obj.headerhandler << SM::WsseAuthHeader.new()
      obj end
```

Security token requirements A security token header must be passed with every API call. This header needs to contain username, password and organization (which is the same information needed in order to login to the Message Studio UI).

The following code is the StrongView security class file (`sm_security.py`) and is required to pass Message Studio API authentication which uses WS-Security SOAP headers.

```
/*
* -------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information                *
*                                                                      *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                      *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                              *
*                                                                      *
* -------------------------------------------------------------------- *
*/

#
# Use header handler to set the authentication headers in each # request.
#
class WsseAuthHeader < SOAP::Header::SimpleHandler
NAMESPACE = 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity- secext-
1.0.xsd'
USERNAME = 'admin@strongview.com'
PASSWORD = 'admin'
ORGANIZATION = 'admin'
SM_SCH = "http://www.strongmail.com/services/2009/03/02/schema"
Header = XSD::QName.new(NAMESPACE, 'Security') OrgToken = XSD::QName.new(SM_SCH,
'OrganizationToken')
def initialize()
super(Header)
```

```
end

#
# Callback method called by soap4r on each request
# to insert the header.
#

def on_simple_outbound
pwd = SOAP::SOAPElement.new(XSD::QName.new(nil, "Password"))
pwd.text = "admin"
pwd.extraattr["Type"] = "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText"
{
  "UsernameToken" => {
  "Username" => USERNAME,
  "Password" => pwd,
  },
  OrgToken => {
    XSD::QName.new(SM_SCH, 'organizationName') => ORGANIZATION, XSD::QName.new(SM_SCH,
'subOrganizationId') => "",
  },
}
end
end
```

To use this code, you will need to edit the file:

1. Make sure USERNAME,PASSWORD, and ORGANIZATION is set to appropriate values in:
   `lib/sm-utils.rb (line#10)`

2. Make sure the SOAP property file contains the following line:
   `client.protocol.http.ssl_config.verify_mode = OpenSSL::SSL::VERIFY_NONE`

Sample Java Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* --------------------------------------------------------------- *
*/

package com.strongmail.webserviceclient.samples
import com.strongmail.webserviceclient.samples.generated.MailingService;
import com.strongmail.webserviceclient.samples.generated.MailingService_Service;
import com.strongmail.webserviceclient.samples.generated.objects.*;

import org.apache.cxf.binding.soap.SoapMessage;
import org.apache.cxf.configuration.jsse.TLSClientParameters;
import org.apache.cxf.endpoint.Client;
import org.apache.cxf.endpoint.ClientImpl;
import org.apache.cxf.endpoint.Endpoint;
import org.apache.cxf.frontend.ClientProxy;
import org.apache.cxf.interceptor.Fault;
import org.apache.cxf.phase.Phase;
import org.apache.cxf.phase.PhaseInterceptor;
import org.apache.cxf.transport.http.HTTPConduit;
import org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor;
import org.apache.ws.security.SOAPConstants;
import org.apache.ws.security.handler.WSHandlerConstants;
import org.apache.ws.security.util.WSSecurityUtil;
import org.apache.ws.security.WSConstants;
import org.apache.ws.security.WSPasswordCallback;

import java.util.Scanner;
import java.util.List;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Map;
import java.util.HashMap;
import java.util.Set;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import java.security.KeyStore;
import java.security.cert.X509Certificate;

import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509TrustManager;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import javax.xml.namespace.QName;
import javax.xml.ws.Binding;
import javax.xml.ws.BindingProvider;
import javax.xml.ws.soap.SOAPBinding;
import javax.xml.soap.SOAPException;
import javax.xml.soap.SOAPMessage;
import com.sun.xml.ws.developer.SchemaValidationFeature;
import org.w3c.dom.Element;
import org.w3c.dom.Text;

public class SampleProgram
{
  public static void main(String[] args) throws Exception
  {

    /* Organization, username, password */
    String organization = "admin"; /* Use "/" to specify suborganization. E.g.
admin/subadmin/onemore */
    String username = "admin";
    String password = "admin";

    /* Initialize the service object with the org, username and password details. */
    SampleProgram sample = new SampleProgram();

    try
    {
      MailingService svc = sample.createService(organization, username, password);

      /* Sample for listing users and roles. */
      describeUserListSample(svc);

      /* Sample for listing templates. */
      describeTemplateListSample(svc);

      /* Sample for listing mailings. */
      describeMailingListSample(svc);
      /* Sample for creating a template. */
      describeTemplateCreateSample(svc);
    }
    catch (Exception e)
    {
      System.out.println("Encountered exception: " + e.getMessage());
      throw e;
    }
  }

  private MailingService createService(String organization,
                                       String username,
                                       String password)
    throws Exception
  {
    URL wsdlURL = new URL
("https://priyanka.mazagoankar/sm/services/mailing/2009/03/02?wsdl");

    QName SERVICE_QNAME = new QName(SERVICE_NAMESPACE, SERVICE_NAME);

    /* Create service with schema validation on */
    MailingService_Service mss = new MailingService_Service(wsdlURL, SERVICE_QNAME);
    SchemaValidationFeature feature = new SchemaValidationFeature();
    MailingService svc = mss.getMailingServicePort(feature);

    /* Set endpoint */
    Endpoint endpoint = ClientProxy.getClient(svc).getEndpoint();
    String endpointAddress = endpoint.getEndpointInfo().getAddress();

    ((BindingProvider)svc).getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_
PROPERTY,
                                                    endpointAddress);

    /* By default do not use MTOM since there is a bug in JAXB library. */
    /* See: https://jaxb.dev.java.net/issues/show_bug.cgi?id=588 */
    Binding binding = ((BindingProvider)svc).getBinding();
    ((SOAPBinding)binding).setMTOMEnabled(false);

    /* Next, create the security headers for authentication. */
    /* WSSE standard username token */
    Map<String, Object> outprops = new HashMap<String, Object>();
    outprops.put(WSHandlerConstants.ACTION, WSHandlerConstants.USERNAME_TOKEN);

    /* Set username and password */
    outprops.put(WSHandlerConstants.USER, username);
    outprops.put(WSHandlerConstants.PASSWORD_TYPE, WSConstants.PW_TEXT);
    outprops.put(WSHandlerConstants.PW_CALLBACK_REF, new ClientPasswordCallbackHandler
(password));

    /* Create client authentication interceptor */
    AuthenticationInterceptor authenticationInterceptor =
      new AuthenticationInterceptor(SERVICE_NAMESPACE_SCHEMA, outprops, organization);
    authenticationInterceptor.setAllowMTOM(false);

    /* Set client to use HTTPS - specify certificate and keystore password (read as JVM
arguments.). */
    ClientImpl client = (ClientImpl)ClientProxy.getClient(svc);
    String keystore = System.getProperty("javax.net.ssl.trustStore");
    String keystorePassword = System.getProperty("javax.net.ssl.trustStorePassword");
    KeyStore ks = KeyStore.getInstance("JKS");
    ks.load(new FileInputStream(keystore), keystorePassword.toCharArray());
    TrustManagerFactory tmf = TrustManagerFactory.getInstance(
      TrustManagerFactory.getDefaultAlgorithm());
    tmf.init(ks);
    TrustManager[] tms = tmf.getTrustManagers();
    TLSClientParameters param = new TLSClientParameters();
    param.setSecureSocketProtocol("SSL");
    param.setTrustManagers(tms);
    HTTPConduit https = (HTTPConduit)client.getConduit();
    https.setTlsClientParameters(param);

    /* Set authetication interceptor into the endpoint in the service object. */
    Endpoint clientEndpoint = client.getEndpoint();
    clientEndpoint.getOutInterceptors().add(authenticationInterceptor);

    return svc;
  }

  private static void describeUserListSample(MailingService svc) throws Exception
  {

    /* List users using the list API with the UserFilter. */
    ListRequest userListRequest = new ListRequest();
    UserFilter userFilter = new UserFilter();
    userFilter.setIsAscending(true);
    userFilter.setPageNumber(0);
    userFilter.setRecordsPerPage(10);
    userFilter.setMaxRecordsPerPage(200);
    userListRequest.setFilter(userFilter);
    ListResponse userListResponse = svc.list(userListRequest);

    /* Get details about each user using the get API call. */
    List<User> users = new ArrayList<User>();
    for (ObjectId id : userListResponse.getObjectId())
    {
      UserId userId = (UserId)id;
      GetRequest getRequest = new GetRequest();
      getRequest.getObjectId().add(userId);
      User user = (User)(svc.get(getRequest).getGetResponse().get(0).getBaseObject());
      users.add(user);
    }

    /* Display all users */
    System.out.println("");
    System.out.println("-----------------------------------------------");
    System.out.println("Users List");
    for (User user : users)
    {
      String roles = "";
      if (user.isIsSuperUser())
      {

        /* User is a super user. */
        roles = "Super User";
      }
      else if (user.isIsAdmin())
      {
        /* User is an admin. */
        roles = "Admin User";
      }
      else
      {
        /* Get the roles assigned to the user from the "access" field. */
        for (User.Access accessInfo : user.getAccess())
        {
          RoleId roleId = accessInfo.getRoleId();
          /* Get details about the role (e.g. its name) using the get API call. */
          GetRequest getRequest = new GetRequest();
          getRequest.getObjectId().add(roleId);
          Role role = (Role)(svc.get(getRequest).getGetResponse().get(0).getBaseObject
());
          roles += role.getName() + " ";
        }
        if (roles == "")
        {
          roles = "<None>";
        }
      }

      System.out.println(user.getFirstName() + " " + user.getLastName() + ". Role(s): " +
roles);
    }
  }

  private static void describeTemplateListSample(MailingService svc) throws Exception
  {
    /* List templates using the list API with the UserFilter. */
    ListRequest templateListRequest = new ListRequest();
    TemplateFilter templateFilter = new TemplateFilter();
    templateFilter.setIsAscending(true);
    templateFilter.setPageNumber(0);
    templateFilter.setRecordsPerPage(10);
    templateFilter.setMaxRecordsPerPage(200);
    templateListRequest.setFilter(templateFilter);
    ListResponse templateListResponse = svc.list(templateListRequest);
    /* Get details about each template using the get API call. */
    List<Template> templates = new ArrayList<Template>();
    for (ObjectId id : templateListResponse.getObjectId())
    {
      TemplateId templateId = (TemplateId)id;
      GetRequest getRequest = new GetRequest();
      getRequest.getObjectId().add(templateId);
      Template template = (Template)(svc.get(getRequest).getGetResponse().get
(0).getBaseObject());
      templates.add(template);
    }
    /* Display all templates */
    System.out.println("");
    System.out.println("-------------------------------------------------");
    System.out.println("Templates List");
    for (Template template : templates)
    {
      System.out.println(template.getName());
    }
  }

  private static void describeMailingListSample(MailingService svc) throws Exception
  {
    /* List batch mailings using the list API with the MailingFilter. */
    ListRequest mailingListRequest = new ListRequest();
    MailingFilter mailingFilter = new MailingFilter();
    mailingFilter.setIsAscending(true);
    mailingFilter.setPageNumber(0);
    mailingFilter.setRecordsPerPage(10);
    mailingFilter.setMaxRecordsPerPage(200);
    mailingListRequest.setFilter(mailingFilter);
    ListResponse mailingListResponse = svc.list(mailingListRequest);
    /* Get details about each mailing using the get API call. */
    List<Mailing> mailings = new ArrayList<Mailing>();
    for (ObjectId id : mailingListResponse.getObjectId())
    {
      MailingId mailingId = (MailingId)id;
      GetRequest getRequest = new GetRequest();
      getRequest.getObjectId().add(mailingId);
      Mailing mailing = (Mailing)(svc.get(getRequest).getGetResponse().get
(0).getBaseObject());
      mailings.add(mailing);
    }
    /* Display all batch mailings */
    System.out.println("");
    System.out.println("------------------------------------------------");
    System.out.println("Mailings List");
    for (Mailing mailing : mailings)
    {
      System.out.println(mailing.getName());
    }
  }

  private static SystemAddressId createAddress(MailingService svc, String type) throws
Exception
  {
    SystemAddress address = new SystemAddress();
    address.setEmailAddress(type+System.currentTimeMillis()+"@sm.com");
    if (type.equalsIgnoreCase("bounce"))
    {
      address.setIsBounce(true);
    }
    else if (type.equalsIgnoreCase("from"))
    {
      address.setIsFrom(true);
    }
      else
    {
      address.setIsReply(true);
    }
    CreateRequest createAddressRequest = new CreateRequest();
    createAddressRequest.getBaseObject().add(address);
    SystemAddressId addressId = (SystemAddressId)(svc.create
(createAddressRequest).getCreateResponse().get(0).getObjectId());
    return addressId;
  }

  private static void describeTemplateCreateSample(MailingService svc) throws Exception
  {
    String name = "Sample_Template_" + System.currentTimeMillis();
    System.out.println("");
    System.out.println("------------------------------------------------");
    System.out.println("Template create");
    System.out.println("");

    Template sampleTemplate = new Template();

    /* Initialize template meta information and envelope properties. */
    sampleTemplate.setName(name);
    sampleTemplate.setDescription("A Sample Template created via the Message Studio API
sample code");
    sampleTemplate.setIsApproved(true);
    sampleTemplate.setBodyEncoding(Encoding.SEVEN_BIT);
    sampleTemplate.setHeaderEncoding(Encoding.SEVEN_BIT);
    sampleTemplate.setOutputBodyCharSet(CharSet.UTF_8);
    sampleTemplate.setOutputHeaderCharSet(CharSet.UTF_8);

    /* Create and set system addresses for the template (you can use existing ones if
needed). */
    sampleTemplate.setBounceAddressId(createAddress(svc, "bounce"));
    sampleTemplate.setFromAddressId(createAddress(svc, "from"));
    sampleTemplate.setReplyAddressId(createAddress(svc, "reply"));

    /* Set template content and related properties. */
    MessagePart templateContentObject = new MessagePart();
    templateContentObject.setFormat(MessageFormat.HTML);
    templateContentObject.setContent("<html><head>Sample Content</head><body>Hi ##first_
name## ##last_name##!<br/>Welcome to our new users program! You can click <a
href=\"http:www.yourdomain.com/newuser\">here</a> to get a tour of our
services</body</html>");
    sampleTemplate.getMessagePart().add(templateContentObject);

    /* Make a call to the create API. */
    CreateRequest createTemplateRequest = new CreateRequest();
    createTemplateRequest.getBaseObject().add(sampleTemplate);
    TemplateId createdTemplateId = (TemplateId)(svc.create
(createTemplateRequest).getCreateResponse().get(0).getObjectId());

    /* Get details about the just created template using the get API call. */
    GetRequest getRequest = new GetRequest();
    getRequest.getObjectId().add(createdTemplateId);
    Template createdTemplate = (Template)(svc.get(getRequest).getGetResponse().get
(0).getBaseObject());

    System.out.println("Created template '" + createdTemplate.getName() + "'");
  }

  /* Service name constant: defined in WSDL */
  private static String SERVICE_NAME = "MailingService";
  private static String SERVICE_NAMESPACE =
"http://www.strongmail.com/services/2009/03/02";
  private static String SERVICE_NAMESPACE_SCHEMA =
"http://www.strongmail.com/services/2009/03/02/schema";

  class ClientPasswordCallbackHandler implements CallbackHandler
  {
    private String password;

    public ClientPasswordCallbackHandler(String password)
    {
      this.password = password;
    }

    public void handle(Callback[] callbacks)
    throws IOException, UnsupportedCallbackException
    {
    WSPasswordCallback pc = (WSPasswordCallback)callbacks[0];
    pc.setPassword(password);
    }
  }

  class AuthenticationInterceptor extends WSS4JOutInterceptor
  {
    /* Schema namespace */
    private String schemaNS;

    /* Interceptor to set headers: organizationName */
    private PhaseInterceptor phaseInterceptor;

    /* organizationName to login */
    private String organizationName;
    private boolean isSSO;
    public AuthenticationInterceptor(String schemaNS,
    Map<String, Object> properties,
    String orgName)
    {
      super(properties);
      this.schemaNS = schemaNS;
      this.phaseInterceptor = new WSS4JOutInterceptorInternal();
      this.organizationName = orgName;
    }

    public void setOrganizationName(String orgName)
    {
      this.organizationName = orgName;
    }

    public void setUsername(String username)
    {
      getProperties().put(WSHandlerConstants.USER, username);
    }

    public void setPassword(String password)
    {
      getProperties().put(WSHandlerConstants.PW_CALLBACK_REF,
      new ClientPasswordCallbackHandler(password));
    }

    public void handleMessage(SoapMessage message) throws Fault
    {
      super.handleMessage(message);
      message.getInterceptorChain().add(phaseInterceptor);
    }

    /**
    * Set SOAP header before sending out message. Header includes username, password,
    * organizationNamed. This message set organizationName and parent class
    * will set username/password.
    */
    class WSS4JOutInterceptorInternal implements PhaseInterceptor<SoapMessage>
    {
      private static final String ORGANIZATION_TOKEN_ELEMENT_NAME = "OrganizationToken";
      private static final String ORGANIZATION_ELEMENT_NAME = "organizationName";

      public WSS4JOutInterceptorInternal()
      {
        super();
      }

      public void handleMessage(SoapMessage message)
      {
        try
        {
          Element securityHeader = getSecurityHeader(message);

          /* Create <OrganizationToken> and <organizationName> element, */
          /* and set <organizationName> as child of <OrganizationToken> */
          Element e = securityHeader.getOwnerDocument().createElementNS(schemaNS,
                                                                        ORGANIZATION_
TOKEN_ELEMENT_NAME);
          Element e2 = securityHeader.getOwnerDocument().createElementNS(schemaNS,
                                                                        ORGANIZATION_
ELEMENT_NAME);
          Text t = securityHeader.getOwnerDocument().createTextNode(
            organizationName);
          e2.appendChild(t);
          e.appendChild(e2);
          securityHeader.appendChild(e);
        }
        catch (Throwable e)
        {
          e.printStackTrace();
        }
      }

      private Element getSecurityHeader(SoapMessage message) throws SOAPException
      {
        SOAPMessage doc = message.getContent(SOAPMessage.class);
        String actor = (String)getOption(WSHandlerConstants.ACTOR);

        SOAPConstants sc = WSSecurityUtil.getSOAPConstants(doc.getSOAPPart
().getDocumentElement());
        return WSSecurityUtil.getSecurityHeader(doc.getSOAPPart(), actor, sc);
      }

      public Set<String> getAfter()
      {
        return Collections.emptySet();
      }

      public Set<String> getBefore()
      {
        return Collections.emptySet();
      }

      public String getId()
      {
        return WSS4JOutInterceptorInternal.class.getName();
      }

      public String getPhase()
      {
        return Phase.POST_PROTOCOL;
      }

      public void handleFault(SoapMessage message)
      {
      }
    }
  }
}
```

Sample C# Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

using System;
using System.Linq;
using System.Text;
using System.Collections.Generic;
using System.Web.Services;
using System.Xml;
using System.IO;
using System.Threading;
using System.Security.Cryptography.X509Certificates;
using Microsoft.Web.Services.Timestamp;
using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Configuration;
using Microsoft.Web.Services2.Security;
using Microsoft.Web.Services2.Security.Tokens;
using Microsoft.Web.Services2.Policy;
using Microsoft.Web.Services2.Referral;

namespace ConsoleApplication1
{
  class SampleProgram
  {

    /* Namespace and endpoint URL: change to your server */
    public const string ns = "http://www.strongmail.com/services/2009/03/02/schema";
    public const string url = "https://127.0.0.1:4443/sm/services/mailing/2009/03/02";

    /* Generated MailingService stub */
    MailingService svc;
    public static void Main(string[] args)
    {

      /* Organization, username, password */
      string organization = "admin"; /* Use "/" to specify suborganization. E.g.
admin/subadmin/onemore */
      string username = "admin";
      string password = "admin";

      /* Initialize the service object with the org, username and password details. */
      MailingService svc = createService(username, password, organization);

      /* Sample for listing users and roles. */
      describeUserListSample(svc);

      /* Sample for listing templates. */
      describeTemplateListSample(svc);

      /* Sample for listing mailings. */
      describeMailingListSample(svc);
      /* Sample for creating a template. */
      describeTemplateCreateSample(svc);
    }
  }

  /* Create a MailingService client with proper headers. */
  private static MailingService createService(string username, string password, string
organization)
  {

    /* Create service and set endpoint */
    MailingService svc = new MailingService();
    svc.Url = url;
    Console.WriteLine("Endpoint: " + svc.Url);

    /* Ignore Certificate by trusting all certificate */
    System.Net.ServicePointManager.CertificatePolicy = new TrustAllCertificatePolicy();

    /* Create security tokens for SOAP header */
    UsernameToken userToken = new UsernameToken(username, password,
PasswordOption.SendPlainText);
    SoapContext requestContext = svc.RequestSoapContext;
    requestContext.Security.Tokens.Add(userToken);

    /* Use filter to set organization token and get rid of timestamp since it does not
work with Java server. */
    MailingServiceOutFilter myfilter = new MailingServiceOutFilter(organization, ns);
    svc.Pipeline.OutputFilters.Insert(0, myfilter);
    return svc;
  }

  private static void describeUserListSample(MailingService svc)
  {

    /* List users using the list API with the UserFilter. */
    ListRequest userListRequest = new ListRequest();
    UserFilter userFilter = new UserFilter();
    userFilter.isAscending = true;
    userFilter.isAscendingSpecified = true;
    userFilter.pageNumber = 0;
    userFilter.pageNumberSpecified = true;
    userFilter.recordsPerPage = 10;
    userFilter.recordsPerPageSpecified = true;
    userListRequest.filter = userFilter;
```

```
ObjectId[] userIds = svc.list(userListRequest).objectId;

/* Get details about each user using the get API call. */
User[] users = new User[userIds.Length];
for (int i = 0; i < userIds.Length; i++)
{
  UserId userId = (UserId)(userIds[i]);
  ObjectId[] objectIds = new ObjectId[1];
  objectIds[0] = userId;
  User user = (User)(svc.get(objectIds).getResponse[0].baseObject);
  users[i] = user;
}

/* Display all users */
Console.WriteLine("");
Console.WriteLine("---------------------------------------------------");
Console.WriteLine("Users List");
for (int i = 0; i < users.Length; i++)
{
  string roles = "";
  if (users[i].isSuperUser)
  {

    /* User is a super user. */
    roles = "Super User";
  }
  else if (users[i].isAdmin)
  {

    /* User is an admin. */
    roles = "Admin User";
  }
  else
  {
    User.Access assignedRoles = users[i].access;

    /* Get the roles assigned to the user from the "access" field. */
    for (int j = 0; j < assignedRoles.Length; j++)
    {
      RoleId roleId = assignedRoles[j].roleId;

      /* Get details about the role (e.g. its name) using the get API call. */
      ObjectId[] objectIds = new ObjectId[1];
      objectIds[0] = roleId;
      Role role = (Role)(svc.get(objectIds).getResponse[0].baseObject);
```

```
          roles += role.getName() + " ";
        }
        if (roles == "")
        {
          roles = "<None>";
        }
      }

      Console.WriteLine(users[i].firstName + " " + users[i].lastName + ". Role(s): " +
roles);
    }
  }

  private static void describeTemplateListSample(MailingService svc)
  {

    /* List users using the list API with the TemplateFilter. */
    ListRequest templateListRequest = new ListRequest();
    TemplateFilter templateFilter = new TemplateFilter();
    templateFilter.isAscending = true;
    templateFilter.isAscendingSpecified = true;
    templateFilter.pageNumber = 0;
    templateFilter.pageNumberSpecified = true;
    templateFilter.recordsPerPage = 10;
    templateFilter.recordsPerPageSpecified = true;
    templateListRequest.filter = templateFilter;
    ListResponse templateListResponse = svc.list(templateListRequest);

    /* Get details about each template using the get API call. */
    ObjectId[] templateIds = svc.list(templateListRequest).objectId;
    Template[] templates = new Template[templateIds.Length];
    for (int i = 0; i < templateIds.Length; i++)
    {

      TemplateId templateId = (TemplateId)(templateIds[i]);
      ObjectId[] objectIds = new ObjectId[1];
      objectIds[0] = templateId;
      Template template = (Template)(svc.get(objectIds).getResponse[0].baseObject);
      templates[i] = template;
    }

    /* Display all templates. */
    Console.WriteLine("");
    Console.WriteLine("--------------------------------------------------");
    Console.WriteLine("Templates List");
    for (int i = 0; i < templates.Length; i++)
    {
      Console.WriteLine(templates[i].name);
    }
  }

  private static void describeMailingListSample(MailingService svc)
  {

    /* List users using the list API with the MailingFilter. */
    ListRequest mailingListRequest = new ListRequest();
    MailingFilter mailingFilter = new MailingFilter();
    mailingFilter.isAscending = true;
    mailingFilter.isAscendingSpecified = true;
    mailingFilter.pageNumber = 0;
    mailingFilter.pageNumberSpecified = true;
    mailingFilter.recordsPerPage = 10;
    mailingFilter.recordsPerPageSpecified = true;
    mailingListRequest.filter = mailingFilter;
    ListResponse mailingListResponse = svc.list(mailingListRequest);

    /* Get details about each mailing using the get API call. */
    ObjectId[] mailingIds = svc.list(mailingListRequest).objectId;
    Mailing[] mailings = new Mailing[mailingIds.Length];
    for (int i = 0; i < mailingIds.Length; i++)
    {
      MailingId mailingId = (mailingId)(mailingIds[i]);
      ObjectId[] objectIds = new ObjectId[1];
      objectIds[0] = mailingId;
      Mailing mailing = (Mailing)(svc.get(objectIds).getResponse[0].baseObject);
      mailings[i] = mailing;
    }

    /* Display all mailings. */
    Console.WriteLine("");
    Console.WriteLine("------------------------------------------------");
    Console.WriteLine("Mailings List");
    for (int i = 0; i < mailings.Length; i++)
    {
      Console.WriteLine(mailings[i].name);
    }
  }

  private static SystemAddressId createAddress(MailingService svc, string type)
  {
    SystemAddress address = new SystemAddress();
    address.emailAddress = type + "@sm.com";
    if (type == "bounce")
    {
      address.isBounce = true;
    }
    else if (type == "from")
    {
      address.isFrom = true;
    }
    else
    {
      address.isReply =true;
    }

    BaseObject[] objects = new BaseObject[1];
    objects[0] = address;
    return (SystemAddressId)(svc.create(objects).createResponse[0].objectId);
  }

  private static void describeTemplateCreateSample(MailingService svc)
  {
    string name = "Sample_Template";
    Console.WriteLine("");
    Console.WriteLine("-------------------------------------------------");
    Console.WriteLine("Template create");
    Console.WriteLine("");
    Template template = new Template();

    /* Initialize template meta information and envelope properties. */
    template.bodyEncoding = Encoding.BASE64;
    template.description = "Template from dotnet";
    template.fromName = "From Name";
    template.headerEncoding = Encoding.EIGHT_BIT;
    template.isApproved = true;
    template.name = templateName;
    template.subject = "template subject from dotnet";
    template.Item = CharSet.ASCII;
    template.Item1 = CharSet.UTF8;

    /* Create and set system addresses for the template (you can use existing ones if
needed). */
    template.bounceAddressId = createAddress(svc, "bounce");
    template.fromAddressId = createAddress(svc, "from");
    template.replyAddressId = createAddress(svc, "reply");
    /* Set template content and related properties. */
    MessagePart messagePart = new MessagePart();
    messagePart.content = "<html><head>Sample Content</head><body>Hi ##first_name##
##last_name##!<br/>Welcome to our new users program! You can click <a
href=\"http:www.yourdomain.com/newuser\">here</a> to get a tour of our
services</body</html>";
    messagePart.format = MessageFormat.HTML;
    messagePart.isXsl = false;
    template.messagePart = new MessagePart[] { messagePart };

    /* Make a call to the create API. */
    BaseObject[] objects = new BaseObject[1];
    objects[0] = template;
    TemplateId createdTemplateId = (TemplateId)(svc.create(objects).createResponse
[0].objectId);

    /* Get details about the just created template using the get API call. */
    ObjectId[] objectIds = new ObjectId[1];
    objectIds[0] = createdTemplateId;
    Template createdTemplate = (Template)(svc.get(objectIds).getResponse[0].baseObject);

    Console.WriteLine("Created template '" + createdTemplate.name + "'");
  }

  /* A policy to trust all certificate */
  public class TrustAllCertificatePolicy : System.Net.ICertificatePolicy
  {
    public TrustAllCertificatePolicy()
    { }
    public bool CheckValidationResult(System.Net.ServicePoint sp,
    X509Certificate cert, System.Net.WebRequest req, int problem)
    {
      return true;
    }
  }

  public class MailingServiceOutFilter : SoapOutputFilter
  {
    private string _ns; /* namespace of organization name */
    private string _organizationName;   /* organization name to login */
    public MailingServiceOutFilter() { }
    public MailingServiceOutFilter(string organizationName, string ns)
    {
      _organizationName = organizationName;
      _ns = ns;
    }

    public override void ProcessMessage(SoapEnvelope envelope)
    {

      /* Add OrganizationName token */
      XmlElement OrgTokenElement = envelope.CreateElement("sm", "OrganizationToken", _
ns);
      XmlElement orgElement = envelope.CreateElement("sm", "organizationName", _ns);

      /* Set organizationName value */
      XmlText orgName = envelope.CreateTextNode(_organizationName);
      orgElement.AppendChild(orgName);
      OrgTokenElement.AppendChild(orgElement);

      /* Remove timestamp token */
      IEnumerator it = envelope.Header.GetEnumerator();
      while(it.MoveNext()){
        XmlElement el = (XmlElement) it.Current;

        if (el.Name.Equals("wsse:Security")) {
          el.AppendChild(OrgTokenElement);

          /* Remove Timestamp */
          XmlNodeList timestamp = el.GetElementsByTagName("wsu:Timestamp");
          el.RemoveChild(timestamp.Item(0));
        }
      }
    }
  }
}
```

XML Examples

The following are examples of the XML requests and responses for the calls shown in the Java and C# sample code. The examples are categorized by the Web Services API Version (1 or 2)

Web Services API v1 Examples
• **Mailing Create Request (v1) on page 55**

Web Services API v1 Examples

Mailing Create Request (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                *
*                                                                    *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                    *
*  Visit http://www.strongview.com for more information             *
*                                                                    *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                    *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                            *
*                                                                    *
* ----------------------------------------------------------------- *
*/
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"
wsu:Id="UsernameToken-20671040">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">admin</wsse:Password>
      </wsse:UsernameToken>
      <OrganizationToken xmlns="http://www.strongmail.com/services/2009/03/02/schema">
        <organizationName>admin</organizationName>
      </OrganizationToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <create xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <baseObject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="StandardMailing">
       <attachmentId>
          <id>3300</id>
       </attachmentId>
       <bodyEncoding>BASE64</bodyEncoding>
        <bounceAddressId>
          <id>98</id>
        </bounceAddressId>
        <isApproved>true</isApproved>
        <isCompliant>true</isCompliant>
        <contentBlockId>
          <id>4100</id>
        </contentBlockId>
        <fieldDelimiter>::</fieldDelimiter>
        <format>HTML</format>
        <fromAddressId>
          <id>97</id>
        </fromAddressId>
        <fromName>From Name</fromName>
        <headerEncoding>EIGHT_BIT</headerEncoding>
        <header>mailing header</header>
        <name>cxf_stmailing12345</name>
        <priority>NORMAL</priority>
        <outputBodyCharSet>UTF-8</outputBodyCharSet>
        <outputHeaderCharSet>ASCII</outputHeaderCharSet>
        <replyAddressId>
          <id>99</id>
        </replyAddressId>
        <rowDelimiter>
        </rowDelimiter>
        <subject>mailing subject</subject>
        <templateId>
          <id>3600</id>
        </templateId>
        <type>RECURRING</type>
        <description>StandardMailing from CXF</description>
        <schedule>
          <startDateTime>2014-04-25T14:06:23.825+05:30</startDateTime>
          <recurrence>
            <endAfterXMailings>100</endAfterXMailings>
            <dailyRecurrence>
              <interval>2</interval>
            </dailyRecurrence>
          </recurrence>
        </schedule>
        <eliminateDuplicates>true</eliminateDuplicates>
        <includedTargetId>
          <id>5700</id>
        </includedTargetId>
        <seedListId>
          <id>3200</id>
        </seedListId>
        <suppressionListId>
          <id>3300</id>
        </suppressionListId>
      </baseObject>
    </create>
  </soap:Body>
</soap:Envelope>
```

Mailing Create Response (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getResponse xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <success>true</success>
      <fault xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
      <createResponse>
        <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="StandardMailingId">
           <id>2200</id>
        </objectId>
      </createResponse>
    </createResponse>
  </soap:Body>
</soap:Envelope>
```

Mailing Get Request (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
```

```
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-14858725">
        <wsse:Username xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">admin</wsse:Username>
        <wsse:Password xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">admin</wsse:Password>
      </wsse:UsernameToken>
      <OrganizationToken xmlns="http://www.strongmail.com/services/2009/03/02/schema">
        <organizationName>admin</organizationName>
      </OrganizationToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <get xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="MailingId">
        <id>100</id>
      </objectId>
    </get>
  </soap:Body>
</soap:Envelope>
```

Mailing Get Response (v1)

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information            *
*                                                                  *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getResponse xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <success>true</success>
      <fault xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <getResponse>
        <baseObject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="StandardMailing">
          <modifiedTime>2012-10-14T20:10:24.798-07:00</modifiedTime>
          <objectId xsi:type="StandardMailingId"><id>101</id></objectId>
          <version>7</version>
          <bodyEncoding>SEVEN_BIT</bodyEncoding>
          <bounceAddressId><id>100</id></bounceAddressId>
          <isApproved>true</isApproved>
          <isCompliant>true</isCompliant>
          <fieldDelimiter>::</fieldDelimiter>
          <format>HTML</format>
          <fromAddressId><id>200</id></fromAddressId>
          <fromName>Dummy From Name</fromName>
          <headerEncoding>SEVEN_BIT</headerEncoding>
          <lastGoodStatus>INITIATING_SAVING</lastGoodStatus>
          <mailingClassId><id>1</id></mailingClassId>
          <name>m1</name>
          <priority>NORMAL</priority>
          <outputBodyCharSet>UTF-8</outputBodyCharSet>
          <outputHeaderCharSet>UTF-8</outputHeaderCharSet>
          <replyAddressId><id>101</id></replyAddressId>
          <rowDelimiter>_sm_row_delim_
          </rowDelimiter>
          <serverErrorCode>0</serverErrorCode>
          <status>COMPLETED</status>
          <subject>Sample template</subject>
          <templateId><id>100</id></templateId>
          <type>ONE_TIME</type>
          <createdTime>2012-10-14T20:09:18.610-07:00</createdTime>
          <description></description>
          <organizationId><id>1</id></organizationId>
          <ownerId><id>1</id></ownerId>
          <eliminateDuplicates>true</eliminateDuplicates>
          <includedTargetId><id>100</id></includedTargetId>
        </baseObject>
      </getResponse>
    </getResponse>
  </soap:Body>
</soap:Envelope>
```

Mailing List Request (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-
14858725">
        <wsse:Username xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">admin</wsse:Username>
        <wsse:Password xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">admin</wsse:Password>
      </wsse:UsernameToken>
      <OrganizationToken xmlns="http://www.strongmail.com/services/2009/03/02/schema">
        <organizationName>admin</organizationName>
      </OrganizationToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <list xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <filter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="MailingFilter">
        <isAscending>true</isAscending>
        <pageNumber>0</pageNumber>
        <recordsPerPage>10</recordsPerPage>
        <maxRecordsPerPage>200</maxRecordsPerPage>
      </filter>
    </list>
  </soap:Body>
</soap:Envelope>
```

Mailing List Response (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <listResponse xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="MailingId">
        <id>101</id>
      </objectId>
      <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="MailingId">
        <id>200</id>
      </objectId>
    </listResponse>
  </soap:Body>
</soap:Envelope>
```

Template Create Request (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                             *
*                                                                 *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                 *
*  Visit http://www.strongview.com for more information           *
*                                                                 *
*  You may incorporate this Source Code in your application only  *
*  you own a valid license to do so from StrongView Systems, Inc  *
*  and the copyright notices are not removed from the source cod  *
*                                                                 *
*  Distributing our source code outside your organization         *
*  is strictly prohibited                                         *
*                                                                 *
* --------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-
14858725">
        <wsse:Username xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">admin</wsse:Username>
        <wsse:Password xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">admin</wsse:Password>
      </wsse:UsernameToken>
      <OrganizationToken xmlns="http://www.strongmail.com/services/2009/03/02/schema">
        <organizationName>admin</organizationName>
      </OrganizationToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <create xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <baseObject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Template">
        <description>A Sample Template created via the Message Studio API sample
code</description>
        <bodyEncoding>SEVEN_BIT</bodyEncoding>
        <bounceAddressId><id>17300</id></bounceAddressId>
        <fromAddressId><id>17301</id></fromAddressId>
        <headerEncoding>SEVEN_BIT</headerEncoding>
        <isApproved>true</isApproved>
        <messagePart>
```

```
          <content>&lt;html>&lt;head>Sample Content&lt;/head>&lt;body>Hi ##first_name##
##last_name##!&lt;br/>Welcome to our new users program! You can click &lt;a
href="http:www.yourdomain.com/newuser">here&lt;/a> to get a tour of our
services&lt;/body&lt;/html></content>
          <format>HTML</format>
          <isXsl>false</isXsl>
        </messagePart>
        <outputBodyCharSet>UTF-8</outputBodyCharSet>
        <outputHeaderCharSet>UTF-8</outputHeaderCharSet>
        <name>Sample_Template_1350498788685</name>
        <replyAddressId><id>17302</id></replyAddressId>
      </baseObject>
    </create>
  </soap:Body>
</soap:Envelope>
```

Template Create Response (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <createResponse xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <success>true</success>
      <fault xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <createResponse>
        <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="TemplateId">
          <id>200</id>
        </objectId>
      </createResponse>
    </createResponse>
  </soap:Body>
</soap:Envelope>
```

Template Get Request (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-
14858725">
        <wsse:Username xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">admin</wsse:Username>
        <wsse:Password xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">admin</wsse:Password>
      </wsse:UsernameToken>
      <OrganizationToken xmlns="http://www.strongmail.com/services/2009/03/02/schema">
        <organizationName>admin</organizationName>
      </OrganizationToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <get xmlns="http://www.strongmail.com/services/2009/03/02/schema">
```

```
      <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="TemplateId">
        <id>100</id>
      </objectId>
    </get>
  </soap:Body>
</soap:Envelope>
```

Template Get Response (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getResponse xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <success>true</success>
      <fault xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <getResponse>
        <baseObject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Template">
          <modifiedTime>2012-10-14T20:08:46.233-07:00</modifiedTime>
          <objectId xsi:type="TemplateId"><id>100</id></objectId>
          <version>8</version>
          <createdTime>2012-10-14T20:05:22.284-07:00</createdTime>
          <bodyEncoding>SEVEN_BIT</bodyEncoding>
          <bounceAddressId><id>100</id></bounceAddressId>
          <fromAddressId><id>200</id></fromAddressId>
          <fromName>Dummy From Name</fromName>
          <headerEncoding>SEVEN_BIT</headerEncoding>
          <isApproved>true</isApproved>
          <messagePart>
            <content>&lt;html>
  &lt;head>
    &lt;title>&lt;/title>
  &lt;/head>
  &lt;body style="background-color: white;">
    &lt;br />
    &lt;p> Hi ##name##! We are glad that you subscribed. You personal info is: ID: ##id##
and Name: ##name##&lt;br />
Content block token personalization - ##cbToken##&lt;br />
Rules personalization - ##SM_RULE_rule1##&lt;br /> &lt;br /> &lt;br />
clicktag - simple link &lt;a href="##ENCLICKTAG##2000##ArgDelimiter##
##ArgDelimiter####ArgDelimiter##www.google.com" target="_blank"> Google &lt;/a> to go to
the search page.&lt;br /> &lt;br />
shortclicktag - link with one param in it &lt;a
href="##ENCLICKTAG##2001##ArgDelimiter####ArgDelimiter####ArgDelimiter##www.yahoo.com?sp=
value"> Yahoo &lt;/a> to go to the search page.&lt;br /> &lt;br />
encodedclicktag - link with some params in it &lt;a
href="##ENCLICKTAG##2002##ArgDelimiter####ArgDelimiter####ArgDelimiter##www.hotmail.com?s
p=v1&amp;amp;ap=v2"> Hotmail &lt;/a> to go to the search page.&lt;/p>
  &lt;/body>
 &lt;/html>
            </content>
            <format>HTML</format>
            <isXsl>false</isXsl>
            <namedLinks>
              <url>www.google.com</url>
              <linkId>2000</linkId>
              <trackingTag>ENCLICKTAG</trackingTag>
            </namedLinks>
            <namedLinks>
              <url>www.yahoo.com?sp=value</url>
              <linkId>2001</linkId>
              <trackingTag>ENCLICKTAG</trackingTag>
            </namedLinks>
            <namedLinks>
              <url>www.hotmail.com?sp=v1&amp;amp;ap=v2</url>
              <linkId>2002</linkId>
              <trackingTag>ENCLICKTAG</trackingTag>
            </namedLinks>
          </messagePart>
          <outputBodyCharSet>UTF-8</outputBodyCharSet>
          <outputHeaderCharSet>UTF-8</outputHeaderCharSet>
          <name>tp1</name>
          <organizationId><id>1</id></organizationId>
          <ownerId><id>1</id></ownerId>
          <replyAddressId><id>101</id></replyAddressId>
          <subject>Sample template</subject>
        </baseObject>
      </getResponse>
    </getResponse>
  </soap:Body>
</soap:Envelope>
```

Template List Request (v1)

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information            *
*                                                                  *
*  You may incorporate this Source Code in your application only   *
*  you own a valid license to do so from StrongView Systems, Inc   *
*  and the copyright notices are not removed from the source cod   *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-
14858725">
        <wsse:Username xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">admin</wsse:Username>
        <wsse:Password xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">admin</wsse:Password>
      </wsse:UsernameToken>
      <OrganizationToken xmlns="http://www.strongmail.com/services/2009/03/02/schema">
        <organizationName>admin</organizationName>
      </OrganizationToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <list xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <filter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="TemplateFilter">
        <isAscending>true</isAscending>
        <pageNumber>0</pageNumber>
        <recordsPerPage>10</recordsPerPage>
        <maxRecordsPerPage>200</maxRecordsPerPage>
      </filter>
    </list>
  </soap:Body>
</soap:Envelope>
```

Template List Response (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <listResponse xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="TemplateId">
        <id>100</id>
      </objectId>
    </listResponse>
  </soap:Body>
</soap:Envelope>
```

User Get Request (v1)

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information            *
*                                                                  *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-
14858725">
        <wsse:Username xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">admin</wsse:Username>
        <wsse:Password xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">admin</wsse:Password>
      </wsse:UsernameToken>
      <OrganizationToken xmlns="http://www.strongmail.com/services/2009/03/02/schema">
        <organizationName>admin</organizationName>
      </OrganizationToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
     <get xmlns="http://www.strongmail.com/services/2009/03/02/schema">
       <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="UserId">
          <id>1</id>
       </objectId>
     </get>
  </soap:Body>
```

```
</soap:Envelope>
```

User Get Response (v1)

```
/*
* ------------------------------------------------------------------ *
*  STRONGVIEW SYSTEMS                                                *
*                                                                    *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                    *
*  Visit http://www.strongview.com for more information             *
*                                                                    *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                    *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                            *
*                                                                    *
* ------------------------------------------------------------------ *
*/
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getResponse xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <success>true</success>
      <fault xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <getResponse>
        <baseObject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="User">
          <modifiedTime>2012-10-14T19:26:52.375-07:00</modifiedTime>
          <objectId xsi:type="UserId"><id>1</id></objectId>
          <version>2</version>
          <description>Default Super User</description>
          <email>admin@strongview.com</email>
          <firstName>Admin</firstName>
          <isAdmin>true</isAdmin>
          <isSuperUser>true</isSuperUser>
          <lastName>Emp</lastName>
          <login>admin</login>
          <organizationId>
            <id>1</id>
          </organizationId>
        </baseObject>
      </getResponse>
    </getResponse>
  </soap:Body>
</soap:Envelope>
```

User List Request (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
      <wsse:UsernameToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-
14858725">
        <wsse:Username xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">admin</wsse:Username>
          <wsse:Password xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">admin</wsse:Password>
        </wsse:UsernameToken>
        <OrganizationToken xmlns="http://www.strongmail.com/services/2009/03/02/schema">
          <organizationName>admin</organizationName>
        </OrganizationToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <list xmlns="http://www.strongmail.com/services/2009/03/02/schema">
```

```
      <filter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="UserFilter">
        <isAscending>true</isAscending>
        <pageNumber>0</pageNumber>
        <recordsPerPage>10</recordsPerPage>
        <maxRecordsPerPage>200</maxRecordsPerPage>
      </filter>
    </list>
  </soap:Body>
</soap:Envelope>
```

User List Response (v1)

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

 User List Response
------------------
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <listResponse xmlns="http://www.strongmail.com/services/2009/03/02/schema">
      <objectId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="UserId">
        <id>1</id>
      </objectId>
    </listResponse>
  </soap:Body>
</soap:Envelope>
```

Reference

This section provides basic information on Message Studio assets, and detailed information on the API operations.

Frequently Asked Questions

What programming languages can I use with the APIs?

If a specific programming language and associated SOAP client library adhere to the SOAP version 1.2 standard as defined by the W3C, then the resulting client applications should work correctly.

What protocols are used for the APIs?

The Message Studio API uses HTTPS (SSL, port 4443) by default while the EIS APIs uses HTTP (port 9000).

Can I use the APIs over a VPN?

Yes, you can use a VPN with the APIs.

Does an API user need to be created in the Message Studio UI?

The user connecting via the API must be created in the MS UI. StrongView recommends creating a dedicated API admin user for each organization, generating a cryptic password, and assigning to the application and then using it exclusively for application.

General Notes

- When Creating Targets, use “Advanced Mode” so you have full control for defining and retrieving the SQL query that the target is based on.
- Cache results returned from the API to reduce the load you place on the API. For example, when managing a transactional message, instead of issuing a `get()` to return the mailing handle every time, put code in place to refresh the mailing handle cache when an error is received from the API.
- For every object/asset, there is an associated `ObjectID`, which is used to identify the object type. For example, for an `InternalDataSource` there is an `InternalDataSourceId`. If you need to identify an object by its ID, use the `get()` operation. You set the `InternalDataSourceId` in the `GetRequest` so the server knows to fetch an IDS, not another type of object. For most of the operations, an `ObjectId` of a certain type is required.

- All operations take some kind of request object and return a response object. If an `ObjectId` is needed to do an operation (such as `cancel()`, `close()`, etc), then you need to create a request first and set the `ObjectId` inside the request. This makes the operations signature consistent.
- Every operation can have only one request.

Known Conditions

- C#/.NET issues - If you are using C#, the generated C# code will need to be modified. Change the following:

  `System.Web.Services.Protocols.SoapHttpClientProtocol`
  to
  `Microsoft.Web.Services2.WebServicesClientProtocol`

  - Replace all instances of typeof(NameValuePair) with typeof(NameValuePair []).
  - Replace the definition of class TxnSendRequest with definition given in the file SchemaChangeForTxnSend.txt (bundled in tarball)
- In Version 1 (2009/03/02), get() API implicitly converts non-supported tag types to CLICKTAG (e.g., View in Mobile or Facebook tags). View in Mobile and Facebook tags are fully supported in Version 2.
- If upgraded from 5.2.X or earlier to version 6.0 and later and are using a PHP library, transactional injections will not function correctly due to the FileInfo property.

  This is not an issue for any other language that has generated and compiled WSDL stubs.

API Assets

Message Studio uses several types of assets for mailings and other system operations. This section provides a brief description of each asset and lists the operations that are supported for each asset.

For descriptions of each of the assets, refer to **Asset Descriptions on page 78**.

For more information about a specific operation, refer to **API Operations on page 80**.

Table 2: Administration/System Operations

Category/Object	Common Operations
Organization	get, list
User	get, list
Role	get, list
MailingClass (Mailing Server Group)	get, list
Campaign	get, list
SystemAddress	get, list
MediaServer	get, list
WebAnalytics	get,list

Table 3: Data Source Operations

Category/Object	Common Operations	Specific Operations
External-Data-Source	create, get, list, update, delete	addRecords, cancelRefreshRecords, exportRecords, getStatistics, refreshRecords, removeRecords
Internal-Data-Source	create, get, list, update, delete, copy	addRecords, dedupeRecords, upsertRecords, exportRecords, getStatistics, importFtpFile, removeRecords, getRecords, getImport, startImport, getImportDetails

Table 4: Recipient Operations

Category/Object	Common Operations	Specific Operations
Target	create, get, list, update, delete, copy	exportRecords
SuppressionList	create, get, list, update, delete, copy	addRecords, exportRecords, removeRecords
SeedList	create, get, list, update, delete, copy	addRecords, exportRecords, removeRecords

Table 5: Content Operations

Category/Object	Common Operations	Specific Operations
Template	create, get, list, update, delete, copy	importMessagePart, test, validateXsl, fetchLinks, personalize
ContentBlock	create, get, list, update, delete, copy	
Attachment	create, get, list, update, delete	importContent
Rule (Dynamic Content)	create, get, list, update, delete, copy	

Table 6: Mailing Operations

Category/Object	Common Operations	Specific Operations
StandardMailing	create, get, list, update, delete, copy, archive	test, launch, pause, resume, cancel, close, getStatistics
TransactionMailing	create, get, list, update, delete, copy, archive	load, pause, resume, cancel, close, getStatistics, getTxnEasInfo, getTxnMailingHandle, txnSend

Asset Descriptions

This section provides a brief description of each asset.

- **Attachment**—A document or other file that is uploaded as an attachments to a specific template or to a mailing.
- **Campaign**—Categorizes different mailings into a single campaign, which can be used when viewing reports for multiple mailings that are similar.
- **Content Block**—Text files that define a series of tokens that are associated with specific content, and may be linked to a recipient’s profile. Content blocks are used to personalize mailings.
- **Data Source**—Database where recipient data is stored. Recipient data may be stored in the internal PostgreSQL database that is included with Message Studio. Or, the data may be stored in an external database (IBM DB2, Microsoft SQL, mySQL, Oracle, PostgreSQL, Sybase, or Teradata).
- **Dynamic Content**—Message Studio includes a rules-based personalization engine. This engine allows you to create rules based on the target list. Using logical comparisons, you create rules where the target field meets the comparison and causes some action to be taken on the template.
- **Mailing Server Group**—Instructs the Email Delivery Server to use a specific set of IP addresses for delivering an email. Before assigning a mailing class with the mailing, make sure you have configured a Virtual Server Group on the StrongView EDS.
- **Media Server**—Message Studio can connect to internal or external media servers that host media (images or videos) required for a specific template. Message Studio can leverage the StrongView tracking server (either onboard or offboard) as a media server.

- **Organization**—An organization is an entity that represents ownership of business assets and data. For example, an organization owns different data sources, message templates, and mailings.
- **Role**—A collection of permissions that is assigned to a user from the Access screen. A role belongs to a single organization, but is available across that organization’s sub-organizations.
- **Seed List**—Email addresses that are not in your data sources. Common uses include sending messages to a test list (such as an internal department list or ISP lists) or for use with StrongDelivery Tools’ Mailbox Monitor.
- **Standard Mailing**—(Batch) Mailings that are sent to a specific target and are launched manually (or through the API). Standard mailings can be set up as one-time mailings or recurring mailings.
  - **One-time mailings**—Associated with one-time events, such as a one-time promotion.
  - **Recurring mailings**—Share a common template, but that are sent out on a regularly scheduled basis, such as a welcome email for a new subscription.
- **Suppression List**—Suppression lists instruct Message Studio to prevent the sending of a mailing to specific domains, email addresses, or accounts that may be included in a target.
- **System Address**—There are three types of system addresses:
  - **Bounce**—The bounce address is the email address associated with the mailbox where receiving EDSs route asynchronous failures.
  - **From**—The from address identifies the sender of the email, by email address and name.
  - **Reply**—The reply address is the email address where the recipient’s EDS should deliver replies to the mailing.
- **Target**—SQL query that identifies the recipients for a specific mailing. A mailing can include one or more targets. You also have the ability to exclude a target from a mailing.
- **Template**—Files that define the format and content of the message and may be designed in HTML or Text.
- **Transactional Mailing**—Mailings based on pre-determined events (such as, when a customer makes a purchase or changes their password). When the event occurs, your Web server makes an API send call to the server. In most cases you create the transactional mailing either through the UI or through the API.

- **User**—There are three types of users:
  - **Super User**—A super user has all available permissions across all organizations.
  - **Administrator**—An administrator has all available permissions within one organization. An admin can also copy templates and content blocks between organizations they "own" and its sub-organizations.
  - **User**—A user's permissions are determined by their role and access privileges, which establishes the user's rights or permissions to view or modify data or take actions.
- **Web Analytics**—Web analytics tags are supported and pre-configured for Coremetrics, Google Analytics, and Omniture.

API Operations

This section provides details on each of the operations available in the API:

- **addRecords() on page 81**
- **cancel() on page 84**
- **cancelRefreshRecords() on page 86**
- **close() on page 87**
- **copy() on page 89**
- **create() on page 92**
- **dedupeRecords() on page 98**
- **delete() on page 101**
- **exportRecords() on page 103**
- **fetchLinks() on page 106**
- **get() on page 110**
- **getRecords() on page 114**
- **getStatistics() on page 119**
- **importContent() on page 122**
- **importMessagePart() on page 124**
- **launch() on page 128**
- **list() on page 130**
- **load() on page 137**
- **pause() on page 139**
- **refreshRecords() on page 141**
- **removeRecords() on page 143**

addRecords()

Use the `addRecords()` operation to add recipient information to one of the supported assets.

Supported Assets

The following list shows the Message Studio assets that are supported for the `addRecords()` operation:

- External data sources
- Internal data sources
- Extension - Catalog
- Program
- Seed list
- Suppression list
- Extension - Standard

Recommendations

- In most cases, you should add records to your external data source directly to your database and not through the API. StrongView does not recommend adding records to your external data source through the API or the UI.
- Before adding records to a data source, you should check the `DataSourceOperationStatus` to make sure it is `IDLE`. Possible status indicators include:
  - `IDLE`—Indicates there is no operation in progress.
  - `IMPORTING`—Indicates someone is adding records to the data source, either through the API or UI.

- `REFRESHING`—Indicates an external data source is currently refreshing the local copy.

Code Sequence

1. `list` *`(filter)`*
2. `addRecords` *`(request)`*

Supported Requests

The `addRecords()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	**Object/Request**
External Data Source	`AddDataSourceRecordsRequest`
Internal Data Source	`AddDataSourceRecordsRequest`
Seed List	`AddSeedListRecordsRequest`
Suppression List	`AddSuppressionListRecordsRequest`
Program	`AddProgramContactRecordsRequest`
Extension - Standard	`AddDataSourceRecordsRequest`
Extension - Catalog	`AddDataSourceRecordsRequest`

Response

`addRecordsResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unrecognizedObjectTypeFault`
- `unexpectedFault`
- `invalidObjectFault`
- `concurrentModificationFault`

Sample Code

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information            *
*                                                                  *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/

/*
* Add records using a file to the datasource created in crea- teIDS().
*
* @param dataSourceId the data source ID where records will be added to.
* @param dataFile the file that will be uploaded
* @throws Exception any exception
*/

public void addRecordsUsingFile(InternalDataSourceId data- SourceId, File dataFile)
throws Exception {

  /* Create request */
  AddDataSourceRecordsRequest addRecordsRequest = new AddDataSourceRecordsRequest();
  addRecordsRequest.setDataSourceId(dataSourceId);
  addRecordsRequest.setDataSourceId(dataSourceId);
  addRecordsRequest.setContainsFieldNames(true);
  addRecordsRequest.setFieldDelimiter(",");

  /* Set data handler to read data from the file */
  InputStream is = new FileInputStream(dataFile);
  DataHandler dh = new DataHandler(new Attach- mentDataSource("text/plain", is));
  addRecordsRequest.setDataSourceRecordsAttachment(dh);

  /* Make call */
  service.addRecords(addRecordsRequest);
}
```

`cancel()`

Use the `cancel()` operation to cancel a mailing that is in progress. Cancelled is a permanent state.

Supported Assets

The following list shows the Message Studio assets that are supported for the `cancel ()` operation:

- Batch mailing
- Transactional mailing

Recommendations

None.

Code Sequence

1. `list`*`(filter)`* -> returns `object_ids`
2. `cancel`*`(request)`*

Supported Requests

The `cancel()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Batch mailing	`CancelRequest`
Transactional mailing	`CancelRequest`

Response

```
cancelResponse
```

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unexpectedFault`

Sample Code

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information            *
*                                                                  *
*  You may incorporate this Source Code in your application only   *
*  you own a valid license to do so from StrongView Systems, Inc   *
*  and the copyright notices are not removed from the source cod   *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/
/*
* Cancel a Standard Mailing. *
* @param mailingId the ID of the Mailing that will be closed
* @return if close operation is successful
* @throws Exception any exception */

public boolean cancel(MailingId mailingId) throws Exception {
  CancelRequest request = new CancelRequest();
  request.setMailingId(mailingId);
  return service.cancel(request).isSuccess();
}
```

`cancelRefreshRecords()`

Use the `cancelRefreshRecords()` operation to cancel a local copy `refresh()`.

Supported Assets

The following list shows the Message Studio assets that are supported for the `cancelRefreshRecords()` operation:

- External data sources

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `cancelRefreshRecords`*`(request)`*

Supported Requests

The `cancelRefreshRecords()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
External data source	`CancelRefreshRecordsRequest`

Response

```
cancelRefreshRecordsResponse
```

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
/*
* Cancel the refresh of External Data Source records. *
* @param edsId the EIS ID to be cancelled.
* @return if cancel is successful
* @throws Exception any exception */

public boolean cancelRefreshRecords(ExternalDataSourceId edsId) throws Exception {
  CancelRefreshRecordsRequest request = new CancelRefreshRecordsRequest();
  request.setDataSourceId(edsId);
  return service.cancelRefreshRecords(request).isSuccess();
}
```

`close()`

Use the `close()` operation to close out a transactional mailing. Closing a transactional mailing puts it into a completed state. Completed is a permanent state.

Supported Assets

The following list shows the Message Studio assets that are supported for the `close ()` operation:

- Batch mailing
- Transactional mailing

Recommendations

None.

Code Sequence

1. `list` *`(filter)`*
2. `close` *`(request)`*

Supported Requests

The `close()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	**Object/Request**
Batch mailing	`CloseRequest`
Transactional mailing	`CloseRequest`

Response

`closeResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`

- `unexpectedFault`

Sample Code

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information            *
*                                                                  *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/

/*
* Close a Transactional Mailing. *
* @param mailingId the ID of the Mailing that will be closed
* @return if close operation is successful
* @throws Exception any exception */

public boolean close(TransactionalMailingId mailingId) throws Exception {
  CloseRequest request = new CloseRequest();
  request.setMailingId(mailingId);
  return service.close(request).isSuccess();
}
```

copy()

Use the `copy()` operation to copy as asset. Copy supports copying a single asset and giving it a new name, or copying a single asset to an organization within the same organization tree. Copying to another organization requires super-user or admin permissions.

Supported Assets

The following list shows the Message Studio assets that are supported for the `copy()` operation:

- Internal data sources
- Target
- Suppression list
- Seed list
- Template
- Extension - Catalog
- Extension - Standard
- Content blocks
- Dynamic content
- Batch mailing
- Transactional mailing

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `copy`*`(request)`*

Supported Requests

The `copy()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	**Object/Request**
Internal data source	`CopyDataSourceReques`
Target	`CopyTargetRequest`
Suppression list	`CopySuppressionListRequest`
Seed list	`CopySeedListRequest`
Template	`CopyTemplateRequest`
Content block	`CopyContentBlockRequest`
Dynamic content	`CopyRuleRequest`

Functional area	Object/Request
Batch mailing	`CopyMailingRequest`
Transactional mailing	`CopyMailingRequest`

Response

`copyResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unrecognizedObjectTypeFault`
- `unexpectedFault`
- `invalidObjectFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information            *
*                                                                   *
*  You may incorporate this Source Code in your application only   *
*  you own a valid license to do so from StrongView Systems, Inc   *
*  and the copyright notices are not removed from the source cod   *
*                                                                   *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/*
* Copy a data source using a new name.
*
```

```
* @param dataSourceId the data source ID that will be copied
* @param newName the name of new data source
* @return the new InternalDataSource ID
* @throws Exception any exception
*/

public InternalDataSourceId copy- DataSource(InternalDataSourceId dataSourceId, String
newName) throws Exception{

  /* Create copy request */
  CopyDataSourceRequest request = new CopyDataSourceRequest();
  request.setFromId(dataSourceId);
  request.setNewName(newName);
  return (InternalDataSourceId) service.copy(request).getObjectId();
}
```

create()

Use the `create()` operation to create any of the supported assets within Message Studio. This is a batch operation that allows you to pass multiple objects in a single call.

Supported Assets

The following list shows the Message Studio assets that are supported for the `create()` operation:

- External Data Source
- Internal Data Source
- Target
- Seed list
- Suppression list
- Template
- (Assigned) Role
- Role
- Media Server
- Media Server Group
- Content block
- Attachment
- Dynamic content (Rules)
- Batch mailing
- Transactional mailing
- Tracking tags
- Organization
- System Address
- User
- StrongTools

- Extension - Catalog
- Extension - Standard

Recommendations

Creating a mailing or template with `isApproved=true`, triggers validation checks. All objects must have all necessary attributes for launching or the server will return a fault.

Note: Content blocks created in the Message Studio SOAP API must be less than 40 MB in size.

Code Sequence

1. instantiate object
2. set parameters
3. `create`*`(array_of_objects)`*

Supported Requests

The `create()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

Note: You must pass in an array of objects for the `create()` operation. The array of objects may be defined as a single asset or as multiple assets to be created.

The following table maps the functional area to the object/request.

Functional area	Request Object	Special Base Object
External Data Source	`CreateRequest`	`ExternalDataSource`
Internal Data Source	`CreateRequest`	`InternalDataSource`
Target	`CreateRequest`	`Target`
Seed list	`CreateRequest`	`SeedList`
Suppression list	`CreateRequest`	`SuppressList`
Template	`CreateRequest`	`Template`
Content blocks	`CreateRequest`	`ContentBlock`
Attachment	`CreateRequest`	`Attachment`
Dynamic content	`CreateRequest`	`Rule`

Functional area	Request Object	Special Base Object
Batch mailing	`CreateRequest`	`StandardMailing`
Transactional mailing	`CreateRequest`	`TransactionalMailing`
Assigned Role	`CreateRequest`	`AssignedRole`
Organization	`CreateRequest`	`Organization`
Media Server	`CreateRequest`	`MediaServer`
System Address	`CreateRequest`	`SystemAddress`
Mailing Server Group	`CreateRequest`	`MailingClass`
Role	`CreateRequest`	`Role`
User	`CreateRequest`	`User`

Response

`BatchCreateResponse`

The `BatchCreateResponse` will include one or more `CreateResponses` depending on the number of base objects you passed into the operation.

Faults

This operation does not throw a fault. Instead, the `FaultDetail` is wrapped in the response. Any faults will be returned in the `BatchCreateResponse`. You will need to check each individual `CreateResponse` for the exception.

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `AuthorizationFaultDetail`
- `InvalidObjectFaultDetail`
- `ObjectNotFoundFaultDetail`
- `UnexpectedFaultDetail`
- `UnrecognizedObjectTypeFaultDetail`

Recommended Defaults

When you set up a mailing, there are several parameters that you will need to define. In Message Studio, some of these parameters have default values. If you are not sure what value to use, you may consider using the defaults listed below.

Templates

Before a mailing can be sent, the template must have the following elements. You can set these elements against the template or against the mailing:

- `bounceAddressID`
- `fromAddressId`
- `fromName`

In addition, you will have to set the following elements; again, these can be set against the template or against the mailing, but must be set. You may want to use the suggested values for the defaults:

- `bodyEncoding`—7 bit
- `headerEncoding`—7 bit
- `outputBodyCharSet`—UTF-8
- `outputHeaderCharSet`—UTF-8

All templates must be approved (`isApproved`) in order for a mailing to launch.

Standard Mailing Recommendations

Before a mailing can be sent, the mailing must have the following elements (plus, the ones mentioned above). You may want to use the suggested values for the defaults:

- `priority`: normal
- `rowDelimiter`: _sm_row_delim_\n
- `fieldDelimiter`: ::

In most cases, you will want to include the `mailingClassId`, which identifies the IP addresses the EDS should use to send the mailing.

Other notes:

- All mailings must include the compliancy check (`isCompliant`) to be launched.
- All mailings must be approved (`isApproved`) to be launched.

Tracking Tags

Support is provided for the following tracking tags:

- NONE
- ENOPENTAG
- OPENTAG
- CLICKTAG
- SHORTCLICKTAG
- ENCLICKTAG
- UNSUBSCRIBETAG
- SHORTUNSUBSCRIBETAG
- ENUNSUBSCRIBETAG
- FTAFTAG
- ENFTAFTAG
- SHARETAG
- ENSHARETAG
- VIEWINBROWSERTAG
- ENVIEWINBROWSERTAG

Transactional Mailing Recommendations

Before a mailing can be loaded, it must have the following elements (plus, the ones mentioned above in the Templates section). You may want to use the suggested values for the defaults:

- `priority`—normal
- `rowDelimiter`—_sm_row_delim_\n
- `fieldDelimiter`—::

Other notes:

- All mailings must include the compliancy check (`isCompliant`) to be loaded.
- All mailings must be approved (`isApproved`) to be loaded.

SMS Mailings

When you create a transactional mailings over SMS, you must set output character set to either ASCII or an ASCII-based character set. SMS does not support any other character sets.

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/*
* Create a Transactional Mailing.
*
* @param mailingName the name of the mailing
* @param templateId the template used for the mailing
* @param targetId the target used for the mailing
* @return TransactionalMailnigId
* @throws Exception any exception
*/

public TransactionalMailingId createTransactionalMailing(String mailingName, TemplateId
templateId, TargetId targetId) throws Exception {
  TransactionalMailing mailing = new TransactionalMailing();
  setMailingCommonFields(mailing, mailingName, bounceId, fromId, replyId, templateId);
  mailing.setDescription("TransactionalMailing from CXF");
  mailing.setType(MailingType.TRANSACTIONAL);
  mailing.setTargetId(targetId);
  mailing.setMessageType(MessageType.EMAIL);

  /* mailing.setMailingConfig(null); */
  /* Read only */
  /* mailing.getRecordStructure(); */
  /* Read only */

  return (TransactionalMailingId) helper.create(mailing);
}

private void setMailingCommonFields(Mailing mailing, String mailingName, SystemAddressId
bounceId, SystemAddressId fromId, SystemAddressId replyId, TemplateId templateId) {
  mailing.setBodyEncoding(Encoding.BASE_64);
  mailing.setBounceAddressId(bounceId);
  mailing.setCampaignId(null);
  mailing.setFieldDelimiter("::");
  mailing.setFromAddressId(fromId);
  mailing.setFromName("From Name");
  mailing.setHeaderEncoding(Encoding.EIGHT_BIT);
  mailing.setIsApproved(true);
  mailing.setIsCompliant(true);
  mailing.setName(mailingName);
  mailing.setOutputBodyCharSet(CharSet.UTF_8);
  mailing.setOutputBodyCharSetToken(null);

  /* can only set either charset or token */
  mailing.setOutputHeaderCharSet(CharSet.ASCII);
  mailing.setOutputHeaderCharSetToken(null);
  mailing.setPriority(MailingPriority.NORMAL);
  mailing.setReplyAddressId(replyId);
  mailing.setRowDelimiter("\n");
  mailing.setSubject("mailing subject");
  mailing.setTemplateId(templateId);
  mailing.getFormat().add(MessageFormat.HTML);
  mailing.getHeader().add("mailing header");
}
```

dedupeRecords()

Use the `dedupeRecords()` operation to remove duplicate recipient records from an internal data source. Choose between one and four fields in the data source, if all the data in the specified fields match, Message Studio removes the record.

Supported Assets

The following list shows the Message Studio assets that are supported for the `dedupeRecords` operation:

- Extension - Catalog
- Extension - Standard
- Internal data sources

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `dedupeRecords`*`(request)`*

Supported Requests

The `dedupeRecords()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Internal data source	`DedupeDataSourceRecordsRequest`

Response

`dedupeRecordsResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`

- unrecognizedObjectTypeFault
- unexpectedFault
- dedupeRecordsResponse

Sample Code

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information            *
*                                                                  *
*  You may incorporate this Source Code in your application only   *
*  you own a valid license to do so from StrongView Systems, Inc   *
*  and the copyright notices are not removed from the source cod   *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/

/*
* Remove duplicate records for a data source based on field "name".
* If duplicate records are found, keep the latest ones. *
* @param idsId the InternalDataSourceId that will be deduped
* @return if dedupe is successful
* @throws Exception any exception */

public boolean dedupeRecords(InternalDataSourceId idsId) throws Exception {
  DedupeDataSourceRecordsRequest request = new DedupeDataSourceRecordsRequest();
  request.setDataSourceId(idsId);
  request.setOption(DataSourceDedupeOption.KEEP_NEWEST);
  request.getMatchField().add("name");
  return service.dedupeRecords(request).isSuccess();
}
```

`delete()`

Use the `delete()` operation to delete one or more assets from the database. This is a batch operation that allows you to pass multiple ids in a single call.

Supported Assets

The following list shows the Message Studio assets that are supported for the `delete()` operation:

- Internal Data Source
- External Data Source
- Target
- Seed list
- Suppression list
- Template
- Extension - Catalog
- Extension - Standard
- Content block
- Attachment StrongTools
- Dynamic content
- Batch mailing
- Transactional mailing
- Organization
- User
- Role
- Assigned Role
- System Address
- Media Server
- Media Server Group

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `delete`*`(array_of_ids)`*

Supported Requests

The `delete()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

Note: You must pass in an array of objects for the `delete()` operation. The array of objects may be defined as a single asset or as multiple assets to be deleted.

The following table maps the functional area to the object/request.

Functional area	Object/Request	Specific Base Object
External data source	`DeleteRequest`	`ExternalDataSourceId`
Internal data source	`DeleteRequest`	`InternalDataSourceId`
Target	`DeleteRequest`	`TargetId`
Seed list	`DeleteRequest`	`SeedListId`
Suppression list	`DeleteRequest`	`SuppressListId`
Template	`DeleteRequest`	`TemplateId`
Content blocks	`DeleteRequest`	`ContentBlockId`
Attachment	`DeleteRequest`	`AttachmentId`
Dynamic content	`DeleteRequest`	`RuleId`
Batch mailing	`DeleteRequest`	`MailingId`
Transactional mailing	`DeleteRequest`	`TransactionalMailingId`

Response

```
BatchDeleteResponse
```

The `BatchDeleteResponse` will include one or more `DeleteResponses` depending on the number of base objects you passed into the operation.

Faults

This operation does not throw a fault. Instead, the `FaultDetail` is wrapped in the response. Any faults will be returned in the `BatchDeleteResponse`. You will need to check each individual `DeleteResponse`for the exception.

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `AuthorizationFaultDetail`
- `ConcurrentModificationFaultDetail`
- `ObjectNotFoundFaultDetail`

- `UnexpectedFaultDetail`
- `UnrecognizedObjectTypeFaultDetail`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information            *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/*
* Delete an object
*
* @param objectId the objectId that will be deleted
* @throws Exception any exception
*/

public void delete(ObjectId objectId) throws Exception {
  List<ObjectId> objectIds = new ArrayList<ObjectId>();
  objectIds.add(objectId);
  delete(objectIds);
}
```

exportRecords()

Use the `exportRecords()` operation to export the contents of the specified asset to file. You must specify a field delimiter and a row delimiter.

Supported Assets

The following list shows the Message Studio assets that are supported for the `exportRecords()` operation:

- Internal data sources
- External data sources
- Target Extension - Catalog
- Extension - Standard
- Seed list
- Suppression list

Recommendations

Before exporting records to a data source, you should check the `DataSourceOperationStatus` to make sure it is `IDLE`; otherwise, only a partial records set will be exported. Possible status indicators include:

- `IDLE`—Indicates there is no operation in progress.
- `IMPORTING`—Indicates someone is adding records to the data source, either through the API or UI.
- `REFRESHING`—Indicates an external data source is currently refreshing the local copy.

Code Sequence

1. `list`*`(filter)`*
2. `exportRecords`*`(request)`*

Supported Requests

The `exportRecords()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Internal data source	`ExportDataSourceRecordsRequest`
External data source	`ExportDataSourceRecordsRequest`

Functional area	Object/Request
Target	`ExportTargetRecordsRequest`
Seed list	`ExportSeedListRecordsRequest`
Suppression list	`ExportSuppressionListRecordsRequest`

Response

`exportRecordsResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unrecognizedObjectTypeFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/*
* Export data source records to a file.
* @param dataSourceId the data source to be exported
* @param toFile the file to be written
* @throws Exception any exception
*/

public void exportRecords(DataSourceId toFile) throws Exception {

  /* Create and fill in request */
  ExportDataSourceRecordsRequest request DataSourceRecordsRequest();
  request.setDataSourceId(dataSourceId);
  request.setFieldDelimiter("|");
  request.setRowDelimiter("\n");

  /* Export regular records. Use true to export malformed records. */

  request.setUseMalformedRecords(false);
  to dataSourceId, File = new Export-

  /* Wait for data source status to be idle before exporting */
  waitForIdle(dataSourceId);

  /* Make call and save data to file */
  ExportRecordsResponse response = service.exportRecords(request);
  InputStream is = response.getData().getInputStream();
  OutputStream os = new FileOutputStream(toFile);
  IOUtils.copy(is, os);
}
```

fetchLinks()

Use the `fetchlinks()` operation to fetch all the URLs from message template or content block content, so that you can insert tracking tags in them.

Supported Assets

The following list shows the Message Studio assets that are supported for the `fetchlinks()` operation:

- Content block
- Template

Recommendations

None.

Code Sequence

1. Initialize the `template` or `contentBlock` object to be sent to the `create()` API, or, for an edit operation, fetch the `template` or `contentBlock` object using `list(filter)`
2. Call `fetchLinks(request)` using the `template` or `contentBlock` object to fetch the list of URLs in the form of `NamedLink` objects.
3. Modify the tracking tags in the URLs and set them in the `namedLinks` property of the `template` or `contentBlock`
4. Save the `template` or `contentBlock`.

Supported Requests

The `fetchlinks()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Template	`NamedLink/fetchLinks()`
Content block	`NamedLink/fetchLinks()`

Response

`FetchLinksResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `AuthorizationFaultDetail`
- `InvalidObjectFaultDetail`
- `ObjectNotFoundFaultDetail`
- `UnexpectedFaultDetail`
- `UnrecognizedObjectTypeFaultDetail`

Sample code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

private List<NamedLink> fetchLinksForTemplate(Template template, MessageFormat format)
throws Exception {
  List<NamedLink> links = new ArrayList<NamedLink>();
  FetchLinksTemplateRequest request = new FetchLinksTemplateRequest();
  request.setTemplate(template); request.setMessageFormat(format);
  FetchLinksResponse response = service.fetchLinks(request);
  List<FetchLinkResponse> individualLinkResponse = response.getFetchLinkResponse();

  if (individualLinkResponse != null && individualLinkResponse.size() 0)
  {
    for (FetchLinkResponse individualLink : individualLinkResponse)
    {
      links.add(individualLink.getNamedLink());
    }
  }

  return links;
}

private List<NamedLink> fetchLinksForContentBlock(ContentBlock contentBlock) throws
Exception {
  List<NamedLink> links = new ArrayList<NamedLink>();
  FetchLinksContentBlockRequest request = new FetchLinksContentBlockRequest();
  request.setContentBlock(contentBlock);
  FetchLinksResponse response = service.fetchLinks(request);
  List<FetchLinkResponse> individualLinkResponse = response.getFetchLinkResponse();

  if (individualLinkResponse != null && individualLinkResponse.size() > 0)
  {
    for (FetchLinkResponse individualLink : individualLinkResponse)
    {
      links.add(individualLink.getNamedLink());
    }
  }
  return links;
}

private void insertTrackingTagsInTemplate() {

/*
* Assuming the template object is ready, in create/edit operation.         *
* The following call will fetch all URLs in the form of named link objects *
* for the HTML message part of this template.                              */

  List<NamedLink> links = this.fetchLinksForTemplate(template, MessageFormat.HTML);

/* By default, each a CLICKTAG will be assigned to each url. The assigment */
/* can be changed as follows. */

  Map<String, TrackingTag> urlToTagMap = new HashMap<String, TrackingTag>();
  urlToTagMap.put("http://www.somedomain.com/link1", TrackingTag.ENCLICKTAG);
  urlToTagMap.put("http://www.somedomain.com/link2", TrackingTag.UNSUBSCRIBETAG);

  for (int i = 0 ; i < links.size() ; i ++)
  {
    if (urlToTagMap.get(links.get(i).getUrl()) != null)
    {
      links.get(i).setTrackingTag(urlToTagMap.get(links.get(i) .getUrl()));
      links.get(i).setName("link" + (i+1));
    }

/* Other properties like web analytics, link name, etc can also be set in the */
/* named link object.  */
  }

/* Assign the updated links to the message part. */
  template.getMessagePart().get(0).getNamedLinks().clear();
  template.getMessagePart().get(0).getNamedLinks().addAll(links);

/* Then save the contentBlock object the edit/create api. */
```

```
}

private void insertTrackingTagsInContentBlock() {

  /* Assuming the contentBlock object is ready, in create/edit operation. */

  List<NamedLink> links = this.fe- tchLinksForContentBlock(contentBlock);

/* By default, each a CLICKTAG will be assigned to each url. The assignment can be */
/* changed as follows. */

  Map<String, TrackingTag> urlToTagMap = new HashMap<String, Track- ingTag>();
  urlToTagMap.put("http://www.somedomain.com/link1", TrackingTag.ENCLICKTAG);
  urlToTagMap.put("http://www.somedomain.com/link2", TrackingTag.UNSUBSCRIBETAG);
  for (int i = 0 ; i < links.size() ; i ++)
  {
    if (urlToTagMap.get(links.get(i).getUrl()) != null)
    {
      links.get(i).setTrackingTag(urlToTagMap.get(links.get(i).getUrl()));
      links.get(i).setName("link" + (i+1));
    }

/* Other properties like web analytics, link name etc can also be set in the named */
/* link object. */

  }

/* Assign the updated links to the message part. */

  contentBlock.getNamedLinks().clear();
  contentBlock.getNamedLinks().addAll(links);

/* Then save the contentBlock object the edit/create api.  */
  }
}
```

get()

Use the `get()` operation to return the `objectID` of a specified asset. In most cases, you will use the `list()` operation to identify a set of assets, and then use the `get()` operation to return the `objectID`.

Supported Assets

The following list shows the Message Studio assets that are supported for the `get()` operation:

- Organization
- User
- Role
- Campaign
- System address
- Media server
- Web analytics
- Mailing class
- External data source
- Internal data source
- Extension - Catalog
- Extension - Standard
- Target
- Seed list
- Suppression list
- Template
- Content block
- Attachment
- Dynamic content
- Batch mailing
- Transactional mailing

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `get`*`(array_of_ids)`*

Supported Requests

The `get()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

Note: You must pass in an array of objects for the `get()` operation. The array of objects may be defined as a single asset or as multiple assets.

The following table maps the functional area to the object/request.

Functional area	Object/Request	Specific Base Object
Organization	`GetRequest`	`OrganizationId`
User	`GetRequest`	`UserId`
Role	`GetRequest`	`RoleId`
Campaign	`GetRequest`	`CampaignId`
System address	`GetRequest`	`SystemAddressId`
Media server	`GetRequest`	`MediaServerId`
Web analytics	`GetRequest`	`WebAnalyticsId`
Mailing class	`GetRequest`	`MailingClassId`
External data source	`GetRequest`	`ExternalDataSourceId`
Internal data source	`GetRequest`	`InternalDataSourceId`
Target	`GetRequest`	`TargetId`
Seed list	`GetRequest`	`SeedListId`
Suppression list	`GetRequest`	`SuppressListId`
Template	`GetRequest`	`TemplateId`
Content blocks	`GetRequest`	`ContentBlockId`
Attachment	`GetRequest`	`AttachmentId`
Dynamic content	`GetRequest`	`RuleId`
Batch mailing	`GetRequest`	`MailingId`
Transactional mailing	`GetRequest`	`TransactionalMailingId`

Response

```
BatchGetResponse
```

The `BatchGetResponse` will include one or more `GetResponses` depending on the number of base objects you passed into the operation.

Faults

This operation does not throw a fault. Instead the `FaultDetail` is wrapped in the response. Any faults will be returned in the `BatchGetResponse`. You will need to check each individual `GetResponse` for the exception.

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `AuthorizationFaultDetail`
- `ObjectNotFoundFaultDetail`
- `UnexpectedFaultDetail`

- `UnrecognizedObjectTypeFaultDetail`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/**
* Get a list of objects based on give IDs. *
* @param objectIds list of IDs that user want to a get
* @return list of BaseObjects
* @throws Exception any exception
*/

public List<BaseObject> get(List<? extends ObjectId> objectIds) throws Exception {

/* Create request and set objectIds */
  GetRequest request = new GetRequest();
  request.getObjectId().addAll(objectIds);

/* Create list of BaseObject with the same size as list of IDs */

  List<BaseObject> result = new ArrayList<BaseObject>(objectIds.size());

/* Make call */

  BatchGetResponse batchGetResponse = service.get(request);

/* Go through each response inside batch response and get the object if it is successful
*/
/* Print out error message if any object is not retrieved successfully */
/* and set the object to null. */

  for (int i = 0; i < objectIds.size(); i++)
  {
    if (batchGetResponse.getSuccess().get(i))
    {
      result.add(batchGetResponse.getGetResponse().get(i).getBaseObject());
    }
    else {
      FaultDetail faultDetail = batchGetResponse.getFault().get(i);
      System.out.println(faultDetail.getFaultCode() + ": " + fault-
Detail.getFaultMessage());
      result.add(null);
    }
  }
  return result;
}
```

getRecords()

Use the `getRecords()` operation to retrieve up to ten records matching a specified match criteria. The API will accept a list of fields to be matched for retrieving the records. Each of the match fields specified should either be the primary key field or an Email Address field or an SMS Address field. The API will support retrieving up to a maximum of 10 records. It will error out if more than 10 records are matched. This API is only supported for internal data sources. The response may contain no records if the match fields do not match any record in the data source. In other words, the presence of the `dataSourceRecords` field is optional in the response.

Supported Assets

The following list shows the Message Studio assets that are supported for the `getRecords()` operation:

- Internal data sources
- Extension - Standard
- Extension - Catalog

Recommendations

None.

Faults

Refer to **Error Handling on page 164** for a description of any faults that may be returned.

The `getRecords` operation does not allow the following scenarios and throws an `InvalidObjectFault` exception:

- The specified data source (`dataSourceId`) is not of type: Internal.
- A non-existent column is specified in the match fields.
- A column specified in the match fields is neither the primary key, nor an email address field nor an SMS address field.
- A column is specified more than once in the match fields.
- The response contains more than 10 records that satisfy the match criteria.

Get Records Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/* Get records using the getRecords API. */

public void getRecord(InternalDataSourceId dataSourceId) throws Exception
{

/* Wait for data source status to be idle before adding */
```

```
  helper.waitForIdle(dataSourceId);

/* Use primary key (id) to fetch the records */

  GetDataSourceRecordsRequest getRecordRequest1 = new GetDataSourceRecordsRequest();
  getRecordRequest1.setDataSourceId(dataSourceId);
  NameValuePair idCol = new NameValuePair();
  idCol.setName("id");
  idCol.setValue("903");
  getRecordRequest1.getMatchFields().add(idCol);
  GetDataSourceRecordsResponse getRecordsResponse1 = (GetDataSourceRecordsResponse)
service.getRecords(getRecordRequest1);
  if (getRecordsResponse1.getDataSourceRecord() != null &&
getRecordsResponse1.getDataSourceRecord().size() > 0)
  {
    for (DataSourceRecord record : getRecordsResponse1.getDataSourceRecord())
    {
      printDataSourceRecord(record);
    }
  }

/* Wait for data source status to be idle before adding */
  helper.waitForIdle(dataSourceId);

/* Use the email field to fetch the records */
  GetDataSourceRecordsRequest getRecordRequest2 = new Get- DataSourceRecordsRequest();
  getRecordRequest2.setDataSourceId(dataSourceId);
  NameValuePair emailCol = new NameValuePair();
  emailCol.setName("email_address");
  emailCol.setValue("email_for_cxf_upsert-updated@cxf.com");
  getRecordRequest2.getMatchFields().add(emailCol);
  GetDataSourceRecordsResponse getRecordsResponse2 = (GetDataSourceRecordsResponse)
service.getRecords(getRecordRequest2);
  if (getRecordsResponse2.getDataSourceRecord() != null &&
getRecordsResponse2.getDataSourceRecord().size() > 0)
  {
    for (DataSourceRecord record : getRecordsResponse2.getDataSourceRecord())
    {
      printDataSourceRecord(record);
    }
  }
}
```

Request Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
/*
* Validate if a template's xsl content is valid.
*
* @param templateId the TemplateId whose XSL content will be validated.
* @return if the content is valid XSL
* @throws Exception any exception
*/

public boolean validateXsl(TemplateId templateId) throws Exception {
  ValidateXslRequest request = new ValidateXslRequest();
  request.setMessageFormat(MessageFormat.HTML);
  request.setTemplateId(templateId);
  return service.validateXsl(request).isValid();
}


<xs:complexType name="GetDataSourceRecordsRequest">
  <xs:complexContent>
    <xs:extension base="tns:GetRecordsRequest">
      <xs:sequence>
        <xs:element name="dataSourceId" type="tns- :DataSourceId"/>
        <xs:element name="matchFields" type="tns- :NameValuePair" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Response Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information            *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
/*
* Validate if a template's xsl content is valid.
*
* @param templateId the TemplateId whose XSL content will be validated.
* @return if the content is valid XSL
* @throws Exception any exception
*/

public boolean validateXsl(TemplateId templateId) throws Exception {
  ValidateXslRequest request = new ValidateXslRequest();
  request.setMessageFormat(MessageFormat.HTML);
  request.setTemplateId(templateId);
  return service.validateXsl(request).isValid();
}


<xs:complexType name="GetDataSourceRecordsResponse">
  <xs:complexContent>
    <xs:extension base="tns:GetRecordsResponse">
      <xs:sequence>
        <xs:element name="dataSourceRecord" type="tns- :DataSourceRecord"
maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
  Faults
```

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

getStatistics()

Use the `getStatistics()` operation to return basic statistics about a particular asset. For example, when you run a `getStatistics()` operation on a data source you can request the following:

- Total number of invalid email addresses
- Total number of malformed records
- Total number of records
- Total number of unsubscribed recipients
- Time of the last data refresh

For a mailing, you can request the following report data:

- Serial number of the launched mailing
- Amount of elapsed time since the mailing launched
- Launch time
- Completion time
- Number of deferred deliveries
- Number of successful deliveries
- Number of failed deliveries
- Number of invalid email addresses included in the target
- Number of sent emails (this successful deliveries + failed deliveries) I number of targeted recipients
- Total number of clicks
- Total number of complaints
- Total number of opens
- Total number of unsubscribes
- number of clicks by unique recipients

- Number of complaints by unique recipients
- Number of opens by unique recipients
- Number of unsubscribes by unique recipients

Supported Assets

The following list shows the Message Studio assets that are supported for the `getStatistics()` operation:

- Internal data sources
- External data sources
- Batch mailing
- Transactional mailing
- Extension - Standard
- Extension - Catalog

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `getStatistics`*`(request)`*

Supported Requests

The `getStatistics()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

Note: You must pass in an array of objects for the `getStatistics()` operation. The array of objects may be defined as a single asset or as multiple assets.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Internal data source	`GetDataSourceStatisticsRequest`
External data source	`GetDataSourceStatisticsRequest`
Batch mailing	`GetMailingStatisticsRequest`
Transactional mailing	`GetMailingStatisticsRequest`
Extension - Catalog	`GetDataSourceStatisticsRequest`
Extension - Standard	`GetDataSourceStatisticsRequest`

Response

`getStatisticsResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unrecognizedObjectTypeFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/**
```

```
* Retrieve and print out a data source.
*
* @param dataSoruceId the data source ID that will be retrieved and printed.
* @throws Exception any exception
*/

public void getAndPrintStatistics(DataSourceId dataSourceId) throws Exception {

  /* Create request */
  GetDataSourceStatisticsRequest request = new GetDataSourceStatisticsRequest();
  request.setDataSourceId(dataSourceId);

  /* Make call */
  GetDataSourceStatisticsResponse response = (GetDataSourceStatisticsResponse)
service.getStatistics(request);
  System.out.println("\nStatistics for data source " + dataSourceId.getId());
  System.out.println("Total records: " + response.ge- tTotalRecords());
  System.out.println("Total invalid records: " + response.getTotalInvalidRecords());
  System.out.println("Total malformed records: " + response.getTotalMalformedRecords());
  System.out.println("Total unsub records: " + response.getTotalUnsubscribedRecords());
}
```

importContent()

Use the `importContent()` operation to import the actual attachment after you create an attachment using the `create()` operation.

Supported Assets

The following list shows the Message Studio assets that are supported for the `importContent()` operation:

- Attachment

Recommendations

None.

Code Sequence

1. `list`*(filter)*
2. `importContent`*(request)*

or

1. `create` *`(array_of_objects)`*
2. `importContent` *`(request)`*

Supported Requests

The `importContent()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Attachment	`ImportAttachmentContentRequest`

Response

`importContentResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unrecognizedObjectTypeFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/**
* Import attachment.
*
* @param attachmentId the AttachmentId that will be imported to.
* @param attachmentFile the attachment file that will be imported.
* @return if the importing operation is successful.
* @throws Exception any exception
*/

public boolean importContent(AttachmentId attachmentId, File attachmentFile) throws
Exception {
  ImportAttachmentContentRequest request = new ImportAttachmentContentRequest();
  request.setAttachmentId(attachmentId);
  request.setFileName(attachmentFile.getName());

/* Set content */

  InputStream is = new FileInputStream(attachmentFile);
  DataHandler dh = new DataHandler(new ByteArrayDataSource(is, "application/octet-
stream"));
  request.setContent(dh);
  return service.importContent(request).isSuccess();
  }
}
```

importMessagePart()

Use the `importMessagePart()` operation to import the actual template content.

Supported Assets

The following list shows the Message Studio assets that are supported for the `importMessagePart()` operation:

- Template

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `importMessagePart`*`(request)`*

or

1. `create` *`(array_of_objects)`*
2. `importMessagePart` *`(request)`*

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/**
* Import @param added to message part using a zip file and media server.   *
* templateId the template that message part will be @param the image files.*
* mediaServerId the media server that is used to store @param zipFile the  *
* zip file that contains HTML and image files. @param folderName the       *
* directory to unzip the zip file * @return if the importing operation is  *
* successful @throws Exception any exception                               *
*/
```

```
public boolean importMessagePart(TemplateId templateId, MediaServerId mediaServerId, File
zipFile, String folderName) throws Exception {

/* Create request */
  ImportMessagePartRequest request = new Import- MessagePartRequest();
  request.setFolderName(folderName);
  request.setIsXsl(false);
  request.setMediaServerId(mediaServerId);
  request.setTemplateId(templateId);

/* Set zip file */
  InputStream is = new FileInputStream(zipFile);
  DataHandler dh = new DataHandler(new ByteArrayDataSource(is, "application/octet-
stream"));
  request.setZipFile(dh);
  return service.importMessagePart(request).isSuccess();
}
```

Supported Requests

The `importMessagePart()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Attachment	`ImportMessagePartRequest`

Response

`importMessagePartResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unexpectedFault`

Sample Code

```
/*
* ---------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                              *
*                                                                  *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                  *
*  Visit http://www.strongview.com for more information           *
*                                                                  *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/

/**
* Import @param added to message part using a zip file and media server.   *
* templateId the template that message part will be @param the image files.*
* mediaServerId the media server that is used to store @param zipFile the  *
* zip file that contains HTML and image files. @param folderName the       *
* directory to unzip the zip file * @return if the importing operation is  *
* successful @throws Exception any exception                               *
*/

public boolean importMessagePart(TemplateId templateId, MediaServerId mediaServerId, File
zipFile, String folderName) throws Exception {

/* Create request */
  ImportMessagePartRequest request = new Import- MessagePartRequest();
  request.setFolderName(folderName);
  request.setIsXsl(false);
  request.setMediaServerId(mediaServerId);
  request.setTemplateId(templateId);

/* Set zip file */
  InputStream is = new FileInputStream(zipFile);
  DataHandler dh = new DataHandler(new ByteArrayDataSource(is, "application/octet-
stream"));
  request.setZipFile(dh);
  return service.importMessagePart(request).isSuccess();
}
```

`launch()`

Use the `launch()` operation to launch an approved mailing.

Supported Assets

The following list shows the Message Studio assets that are supported for the launch ()operation:

- Batch mailing

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `launch`*`(request)`*

Supported Requests

The `launch()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Batch mailing	`LaunchRequest`

Response

`launchResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/*
* Launch a Standard Mailng.                                            *
* @param mailingId the ID of Standard Mailing that will be launched    *
* @return if launch operation is successful                            *
* @throws Exception any exception                                      *
*/
public boolean launchMailing(StandardMailingId mailingId) throws Exception {
  LaunchRequest request = new LaunchRequest();
  request.setMailingId(mailingId);
  return service.launch(request).isSuccess();
}
```

`list()`

Use the `list()` operation to query Message Studio for a list of ids associated with a specific or series of specific assets. The `list()` operation requires you to set a filter that will return the ids of specific assets. You will basically set a series of conditions that you expect the returned list to conform to; for example, the sample code below searches all system addresses for a specific email address.

Supported Assets

The following list shows the Message Studio assets that are supported for the `list()` operation:

- Organization
- User
- Role
- Campaign
- System address
- Media server
- Web analytics
- Mailing Server Group
- External Data Source
- Internal Data Source
- Extension - Catalog
- Extension - Standard
- Target
- Seed list
- Suppression List
- Template
- Content block
- Attachment
- Dynamic content
- Batch mailing
- Transactional mailing
- Program
- Assigned Role
- StrongTools

Recommendations

Each `listRequest` takes a filter and there is one filter for each object type. Then, you can set one or more conditions for each filter. There are two condition types:

- **Scalar**—Used for a single value
- **Array**—Used for a list of values, such as mailing types, or mailing status

Each condition has an operator, such as `EQUAL` and a value. Based on the condition, a value may be a string, ID, integer, or boolean. In one filter, you can set multiple conditions.

This may include page number, number of records per page, ordering type, or order by fields for each filter. If you do not choose to set these, the default values will be `pagenumber=0` (first page), `number of records=10` per page, and ascending order. The max number of records you can set is 200.

Code Sequence

1. `list`*`(filter)`*

Supported Requests

The `list()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request	Specific Base Object
Organization	`ListRequest`	`OrganizationId`
User	`ListRequest`	`UserId`
Role	`ListRequest`	`RoleFilter`
Campaign	`ListRequest`	`CampaignFilter`
System address	`ListRequest`	`SystemAddressFilter`
Media server	`ListRequest`	`MediaServerFilter`
Web analytics	`ListRequest`	`WebAnalyticsFilter`
Mailing Server Group	`ListRequest`	`MailingClassFilter`
External data source	`ListRequest`	`DataSourceFilter`
Internal data source	`ListRequest`	`DataSourceFilter`
Target	`ListRequest`	`TargetFilter`
Seed list	`ListRequest`	`SeedListFilter`
Suppression list	`ListRequest`	`SuppressListFilter`
Template	`ListRequest`	`TemplateFilter`
Content blocks	`ListRequest`	`ContentBlockFilter`
Attachment	`ListRequest`	`AttachmentFilter`
Dynamic Content	`ListRequest`	`RuleFilter`
Batch mailing	`ListRequest`	`MailingFilter`

Functional area	Object/Request	Specific Base Object
Transactional mailing	`ListRequest`	`MailingFilter`
Program	`ListRequest`	`ProgramFilter`
Assigned Role	`ListRequest`	`AssignedRoleFilter`

Response

`listResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `unrecognizedObjectTypeFault`
- `unexpectedFault`
- `invalidObjectFault`

Sample Code

Sample 1: Filter system addresses by type and a specific email address

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/*
* List SystemAddress by type (bounce/from/reply) and email address.    *
* There is no filter by email address, so we need to get all address   *
* for one type and filter them again.                              *
*                                                                  *
* @param type the SystemAddressType to filter on                   *
* @param emailAddress the email address to match                   *
* @return the SystemAddress that belongs to the type and matches    *
* the email address                                                *
* @throws Exception any exception                                  *
*/

public SystemAddress listByEmailAddress(SystemAddressType type, String emailAddress)
throws Exception {

/* Create and set attribute for filter  */

  SystemAddressFilter filter = new SystemAddressFilter();
  helper.setFilter(filter);

/* Create type condition */

  ScalarStringFilterCondition typeCondition = new ScalarStringFilterCondition();
  typeCondition.setOperator(FilterStringScalarOperator.EQUAL);

/* Set type condition value */

  if (type.equals(SystemAddressType.BOUNCE))
    {
    typeCondition.setValue(SystemAddressType.BOUNCE.value());
    }
  else if (type.equals(SystemAddressType.FROM))
    {
      typeCondition.setValue(SystemAddressType.FROM.value());
    }
    else if (type.equals(SystemAddressType.REPLY))
    {
      typeCondition.setValue(SystemAddressType.REPLY.value());
    }
    filter.setTypeCondition(typeCondition);

/* Create request and make call */
    ListRequest request = new ListRequest();
    request.setFilter(filter);
    List<ObjectId> addressIds = service.list(request).getObjectId();

/* Go through all addresses to get the one that matches email address. */
```

```
    for (ObjectId each: addressIds)
    {
      SystemAddress address = (SystemAddress) helper.get((SystemAddressId) each);
      if (address.getEmailAddress().equals(emailAddress))
      {
          return address;
      }
    }
    return null;
  }
}
```

Sample 2: Lists mailings by name and returns one mailing if the name is matched

```
/*
* ------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information                *
*                                                                      *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                      *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                              *
*                                                                      *
* -------------------------------------------------------------------- *
*/

/*
* List mailings by name: this should return one mailng if name is matched.
*
* @param name the mailing name to be listed
* @throws Exception any exception
*/

public void listByName(String name) throws Exception {
  MailingFilter filter = new MailingFilter();
  helper.setFilter(filter);

/* Create name condition */
```

```
  ScalarStringFilterCondition nameCondition = new ScalarStringFilterCondition();
  nameCondition.setOperator(FilterStringScalarOperator.EQUAL);
  nameCondition.setValue(name);
  filter.setNameCondition(nameCondition);

/* Set the filter to request */
  ListRequest request = new ListRequest();
  request.setFilter(filter);

/* Make call and print results */
  printListResponse(service.list(request));
}
```

Sample 3: Lists mailings in a specific organization

```
/*
* ------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information                *
*                                                                      *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                      *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                              *
*                                                                      *
* -------------------------------------------------------------------- *
*/

/**
* List maiilng in an organization: this can be done using an empty filter.
* The organization is the one from header.
* @throws Exception any exception
*/
public void listByOrganization() throws Exception {

/* Create empty filter: no conditions */

  MailingFilter filter = new MailingFilter(); helper.setFilter(filter);

/* Set the filter to request */
```

```
  ListRequest request = new ListRequest();
  request.setFilter(filter);

/* Make call and print results */

  printListResponse(service.list(request));
}
```

Sample 4: Lists mailings by type and status

```
/*
* ------------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                                  *
*                                                                      *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved.      *
*                                                                      *
*  Visit http://www.strongview.com for more information                *
*                                                                      *
*  You may incorporate this Source Code in your application only if    *
*  you own a valid license to do so from StrongView Systems, Inc.      *
*  and the copyright notices are not removed from the source code.     *
*                                                                      *
*  Distributing our source code outside your organization              *
*  is strictly prohibited                                              *
*                                                                      *
* -------------------------------------------------------------------- *
*/

/**
* List mailings by mailing type and status.
*
* @throws Exception any exception
*/

public void listByTypeAndStatus() throws Exception {

/* Create a mailing filter and set attribute and order by list */
  MailingFilter filter = new MailingFilter(); helper.setFilter(filter); filter.getOrderBy
().add(MailingOrderBy.NAME); filter.getOrderBy().add(MailingOrderBy.MODIFIED_TIME);

/* Create status condition: status in (ACTIVE, EDITIN, COMPLETED) */
  ArrayStringFilterCondition statusCondition = new ArrayStringFilterCondition();
  statusCondition.setOperator(FilterArrayOperator.IN);
  statusCondition.getValue().add(MailingStatus.ACTIVE.value());
  statusCondition.getValue().add(MailingStatus.EDITING.value());
  statusCondition.getValue().add(MailingStatus.COMPLETED.value());
  filter.setStatusCondition(statusCondition);

/* Create Type Condition: type in (RECURRING, TRANSACTIONAL) */
  ArrayStringFilterCondition typeCondition = new ArrayStringFilterCondition();
  typeCondition.setOperator(FilterArrayOperator.IN);
  typeCondition.getValue().add(MailingType.RECURRING.value());
  typeCondition.getValue().add(MailingType.TRANSACTIONAL.value());
  filter.setTypeCondition(typeCondition);

/* Set the filter to request */
  ListRequest request = new ListRequest();
  request.setFilter(filter);

/* Make call and print results */
  printListResponse(service.list(request));
  }

private void printListResponse(ListResponse response) throws Exception {
  List<ObjectId> mailingIds = response.getObjectId();
  System.out.println("\nList Response:");
  for (ObjectId each: mailingIds) {
    Mailing mailing = (Mailing) helper.get(each);
    System.out.println("\t" + each.getId() + " " + mail- ing.getName());
  }
 }
}
```

load()

Use the `load()` operation to load an approved transactional mailing.

Supported Assets

The following list shows the Message Studio assets that are supported for the `load()` operation:

- Transactional mailing

Recommendations

None.

Code Sequence

1. `list(filter)`
2. `load(request)`

Supported Requests

The *`load()`* operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Transactional mailing	`loadRequest`

Response

`loadResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                  *
*  Distributing our source code outside your organization          *
*  is strictly prohibited                                          *
*                                                                  *
* ---------------------------------------------------------------- *
*/

/**
* Load a Transactional Mailing.
* @param mailingId the ID of Transactional Mailing that will be loaded
* @return if load operation is successful
* @throws Exception any exception
*/
public boolean loadMailing(TransactionalMailingId mailingId) throws Exception {
  LoadRequest request = new LoadRequest();
  request.setMailingId(mailingId);
  return service.load(request).isSuccess();
}
```

pause()

Use the `pause()` operation to pause a standard or transactional mailing. Pausing allows you to update template, content, or headers. You cannot modify the schema when you pause a mailing. Paused mailings can be resumed.

Supported Assets

The following list shows the Message Studio assets that are supported for the `pause()` operation:

- Batch mailing
- Transactional mailing

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `pause`*`(request)`*

Supported Requests

The `pause()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Batch mailing	`PauseRequest`
Transactional mailing	`PauseRequest`

Response

`pauseResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
```

```
*/

/**
* Pause a standard or transactional mailing. *
* @param mailingId the ID of the Mailing that will be paused
* @return if close operation is successful
* @throws Exception any exception */

public boolean pause(MailingId mailingId) throws Exception {
  PauseRequest request = new PauseRequest();
  request.setMailingId(mailingId);
  return service.pause(request).isSuccess();
}
```

`refreshRecords()`

Use the `refreshRecords()` operation to refresh the local copy of an external data source.

Supported Assets

The following list shows the Message Studio assets that are supported for the `refreshRecords()` operation:

- External Data Source

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `refreshRecords`*`(request)`*

Supported Requests

The `refreshRecords()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
External data source	RefreshRecordsRequest

Response

refreshRecordsResponse

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- objectNotFoundFault
- authorizationFault
- unexpectedFault
- concurrentModificationFault

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/**
* Refresh External Data Source records.
*
* @param edsId the EIS ID to be refreshed.
* @return if refresh is successful
* @throws Exception any exception
```

```
*/

public boolean refreshRecords(ExternalDataSourceId edsId) throws Exception {
  RefreshRecordsRequest request = new RefreshRecordsRequest();
  request.setDataSourceId(edsId);
  return service.refreshRecords(request).isSuccess();
}
```

`removeRecords()`

Use the `removeRecords()` operation to remove recipient data from a data source, seed list, or suppression list.

Supported Assets

The following list shows the Message Studio assets that are supported for the `removeRecords()` operation:

- Internal data source
- External data source
- Extension - Catalog
- Seed list
- Suppression list
- Extension - Standard

Recommendations

In most cases, you should delete records from your external data source directly from your database and not through the API. StrongMail does not recommend deleting records from your external data source through the API or the UI.

Code Sequence

1. `list`*`(filter)`*
2. `removeRecords`*`(request)`*

Supported Requests

The `removeRecords()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Internal data source	`RemoveDataSourceRecordsRequest`
External data source	`RemoveDataSourceRecordsRequest`
Seed list	`RemoveSeedListRecordsRequest`
Suppression list	`RemoveSuppressionListRequest`

Response

`removeRecordsResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unrecognizedObjectTypeFault`
- `unexpectedFault`
- `concurrentModificationFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/**
```

```
* Remove one record that was created in addRecordsInline() from the SeedList.
*
* @param seedListId the SeedListId whose record will be removed.
* @return the number of record that was removed.
* @throws Exception any exception
*/

public int removeRecordsInline(SeedListId seedListId) throws Exception {
  RemoveSeedListRecordsRequest request = new RemoveSeedListRecordsRequest();
  request.setSeedListId(seedListId);
  request.getAddress().add("seed1@seed.cxf.com");
  return service.removeRecords(request).getRecordsRemoved();
}
```

`resume()`

Use the `resume()` operation to resume delivery of a paused mailing. If you made modifications to the mailing content or headers, the modifications are used when the mailing is resumed.

Supported Assets

The following list shows the Message Studio assets that are supported for the `resume()` operation:

- Batch mailing
- Transactional mailing

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `resume`*`(request)`*

Supported Requests

The `resume()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Batch mailing	`ResumeRequest`
Transactional mailing	`ResumeRequest`

Response

`resumeResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
```

```
*/

/**
* Resume a standard or transactional mailing. *
* @param mailingId the ID of Mailing that will be closed
* @return if close operation is successful
* @throws Exception any exception */

public boolean resume(MailingId mailingId) throws Exception {
  ResumeRequest request = new ResumeRequest();
  request.setMailingId(mailingId);
  return service.resume(request).isSuccess();
}
```

schedule()

Use the `schedule()` operation to schedule a mailing to launch at a specific interval.

Supported Assets

The following list shows the Message Studio assets that are supported for the `schedule()` operation:

- Batch mailing

Recommendations

None.

Code Sequence

1. `create`*`(mailing)`* — Set the schedule in the mailing while saving it
2. `schedule`*`(request)`*

Supported Requests

The `schedule()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Batch mailing	`schedule()`

Response

`ScheduleResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `AuthorizationFaultDetail`
- `InvalidObjectFaultDetail`
- `ObjectNotFoundFaultDetail`
- `UnexpectedFaultDetail`
- `UnrecognizedObjectTypeFaultDetail`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

mailing.setType(MailingType.RECURRING);

/* Set schedule: daily recurrence with every two days */
```

```
SchedulableMailing.Schedule.Recurrence.DailyRecurrence daily = new
SchedulableMailing.Schedule.Recurrence.DailyRecurrence();
daily.setInterval(2);
SchedulableMailing.Schedule.Recurrence recurrence = new
SchedulableMailing.Schedule.Recurrence();
recurrence.setEndAfterXMailings(100); /* launch 100 mailing */
recurrence.setDailyRecurrence(daily);
SchedulableMailing.Schedule schedule = new SchedulableMailing.Schedule();
schedule.setRecurrence(recurrence); /* one hour later */
schedule.setStartDateTime(new Date(System.currentTimeMillis() + 3600000));
mailing.setSchedule(schedule);
...
/* Now call the create mailing API to save the mailing and collect the */
/* mailingId from the create API.*/
/* Finally, schedule the mailing using its mailing id.*/
ScheduleRequest request = new ScheduleRequest();
request.setMailingId(mailingId);
service.schedule(request);
```

send()

Important: The `send()` operation is deprecated. Use **`txnsend()`** call to trigger the sending of messages for a transactional mailing. The `txnsend()` operation provides better send performance.

test()

Use the `test()` operation to test the dynamic content within a template or mailing by sending a test mailing.

Supported Assets

The following list shows the Message Studio assets that are supported for the `test()` operation:

- Batch mailing
- Template

Recommendations

None.

Code Sequence

1. `list` *`(filter)`*
2. `test` *`(request)`*

Supported Requests

The `test()` operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Template	`TestTemplateRequest`
Batch mailing	`TestMailingRequest`

Response

`testResponse`

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unrecognizedObjectTypeFault`
- `unexpectedFault`
- `invalidObjectFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

/**
* Test template for it dynamic content.
* Email will be sent and tokens in the template will be replaced with values.
*
* @param templateId the TemplateId that will be sent out.
* @return if the test request is sent successfully.
* @throws Exception any exception
*/

public boolean testTemplate(TemplateId templateId) throws Exception {
  TestTemplateRequest request = new TestTemplateRequest();
  request.setSubjectPrefix("Subject Prefix: ");
  request.setTemplateId(templateId);
  request.setTestListId(null);
  request.setUseMultiPart(false);
  request.getAddress().add("test_address@testtemplate.com");
  request.getFormat().add(MessageFormat.HTML);

/* Make sure you have these two tokens in your template */
  NameValuePair pair1 = new NameValuePair(); pair1.setName("EMAIL");
  pair1.setValue("testemail@testtemplate.com");
  request.getTokenValue().add(pair1);
  NameValuePair pair2 = new NameValuePair();
  pair2.setName("NAME");
  pair2.setValue("testemail name");
  request.getTokenValue().add(pair2);
  return service.test(request).isSuccess();
}
```

`txnSend()`

There are two methods that can be used for sending transactional messages:

- `txnSend()`—Use this operation for moderate volume/frequency transactional mailing applications. The frequency of `txnSend()` requests should not exceed 500,000 per hour.
- `send()`—Use this API for very high volume/frequency applications after you create the transactional mailings through the Message Studio API operation **create() on page 92**.

Note: `send()`—This method is still supported but has been deprecated due to performance constraints.

Supported Assets

The following list shows the Message Studio assets that are supported for the `txnSend()` operation:

- Transactional mailing

Recommendations

For Message Studio 4.x or earlier, the maximum supported size of an API call is 64 KB, including any database records and message content.

In version 5.0 or greater, the maximum recommended size is 256 KB. Usually this is not a problem, since most transactional API calls will fall below this level.

Code Sequence

1. `list`*`(filter)`*
2. `txnSend`*`(request)`*

Sample Code

The following method uses name/value pairs, which take a string that contains all data to send the transactional email.

```
/*
```

```
* --------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                             *
*                                                                 *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                 *
*  Visit http://www.strongview.com for more information           *
*                                                                 *
*  You may incorporate this Source Code in your application only  *
*  you own a valid license to do so from StrongView Systems, Inc  *
*  and the copyright notices are not removed from the source cod  *
*                                                                 *
*  Distributing our source code outside your organization         *
*  is strictly prohibited                                         *
*                                                                 *
* --------------------------------------------------------------- *
*/
```

Supported Requests

The `txnSend()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Transactional mailing	`TxnSendRequest/TransactionalMailingId)`

Response

boolean success

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

    BadHandleFault

update()

Use the `update()` operation to edit and update the content of the specified asset. For example, you could do any of the following (and more):

- Rename an asset
- Change a target’s query
- Modify a content block

Supported Assets

The following list shows the Message Studio assets that are supported for the `update()` operation:

- Internal Data Sources
- External Data Sources
- Target
- Seed list
- Suppression list
- Template
- Extension - Catalog
- Extension - Standard
- Batch mailing
- Dynamic Content
- Content blocks
- Attachment
- Organization
- User
- Role
- System Address
- Media Server
- Media Server Group
- StrongTools
- Transactional mailing

Recommendations

- Updating a mailing or template to `isApproved=true`, triggers validation checks. All objects must have all necessary attributes for launching or the server will return a fault.
- The following table lists the fields that are read-only fields and cannot be updated:

Asset	Read-only fields
Data source	`objectId`, `createdTime`, `operationStatus`, `modifiedTime` (from BaseObject), `version` (from BaseObject), `organizationId`, `ownerId`.
Target	`objectId`, `totalRecords`, `retargetingDataSourceId`, `createdTime`, `modifiedTime` (from BaseObject), `version` (from BaseObject),`organizationId`, `ownerId`.

Asset	Read-only fields
Seed list	`objectId`, `totalRecords`, `createdTime`, `modifiedTime` (from BaseObject), `version` (from BaseObject), `organizationId`, `ownerId`
Suppression list	`objectId`, `totalRecords`, `createdTime`, `modifiedTime` (from BaseObject), `version` (from BaseObject), `organizationId`, `ownerId`
Template	`objectId`, `createdTime`, `modifiedTime` (from BaseObject), `version` (from BaseObject), `organizationId`, `ownerId`(for Channel mailings)
Content block	`objectId`, `size`, `createdTime`, `modifiedTime` BaseObject), `version` (from BaseObject), `organizationId`, `ownerId`
Attachment	`objectId`, `size`, `createdTime`, `modifiedTime` BaseObject), `version` (from BaseObject), `organizationId`, `ownerId`
Rule	`objectId`, `createdTime`, `modifiedTime` (from BaseObject), `version` (from BaseObject), `organizationId`, ownerId
Batch, and Transactional mailings	`objectId`, `createdTime`, `modifiedTime` (from BaseObject), `version` (from BaseObject), `organizationId`, `ownerId`, `format`, `parentId`, `lastGoodStatus`, `serverErrorCode`, `serverErrorMessage`, `status`for Channel mailings)

Code Sequence

1. `list`*`(filter)`*
2. `get` *`(object_ids)`*
3. `update`*`(array_of_objects)`*

Supported Requests

The *`update()`* operation supports multiple objects that can be passed in as arguments. Each object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

Note: You must pass in an array of objects for the `update()` operation. The array of objects may be defined as a single asset or as multiple assets to be updated.

The following table maps the functional area to the object/request.

Functional area	Object/Request	Specific Base Object
External data source	`updateRequest`	`ExternalDataSource`
Internal data	`updateRequest`	`InternalDataSource`

Functional area	Object/Request	Specific Base Object
source		
Target	updateRequest	Target
Seed list	updateRequest	SeedList
Suppression list	updateRequest	SuppressList
Template	updateRequest	Template
Content blocks	updateRequest	ContentBlock
Attachment	updateRequest	Attachment
Rule	updateRequest	Rule
Batch mailing	updateRequest	StandardMailing
Transactional mailing	updateRequest	TransactionalMailing

Response

`BatchUpdateResponse`

The `BatchUpdateResponse` will include one or more `UpdateResponse`s depending on the number of base objects you passed into the operation.

Faults

This operation does not throw a fault. Instead,s the `FaultDetail` is wrapped in the response. Any faults will be returned in the `BatchUpdateResponse`. You will need to check each individual `UpdateResponse` for the exception.

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `AuthorizationFaultDetail`
- `InvalidObjectFaultDetail`
- `ObjectNotFoundFaultDetail`
- `StaleObjectFaultDetail`
- `UnexpectedFaultDetail`
- `UnrecognizedObjectTypeFaultDetail`

Tracking Tags

Support is provided for the following tracking tags:

- None
- OPENTAG
- ENOPENTAG
- CLICKTAG
- SHORTCLICKTAG
- ENCLICKTAG
- UNSUBSCRIBETAG
- SHORTUNSUBSCRIBETAG l ENUNSUBSCRIBETAG
- FTAFTAG
- ENFTAFTAG
- SHARETAG
- ENSHARETAG
- VIEWINBROWSERTAG
- ENVIEWINBROWSERTAG

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
```

```
/**
* Update an data source with new name.
*
* @param dataSourceId the data source ID whose name will be updated
* @param newName the new name for the data source
* @throws Exception any exception
*/
public void updateDataSourceName(DataSourceId dataSourceId, String newName) throws
Exception {

/* Get the data source first */
  List<ObjectId> dataSourceIds = new ArrayList<ObjectId>(1);
  dataSourceIds.add(dataSourceId);
  DataSource dataSource = (DataSource) help- er.get(dataSourceIds).get(0);

/* Change name and update */
  dataSource.setName(newName);
  helper.update(dataSource);
}
```

upsertRecord()

Use the `upsertRecord()` operation to do a single record add or update in one step.

Important: This API operation is only supported for Internal Data Sources.

Supported Assets

The following list shows the Message Studio assets that are supported for the `upsertRecord()` operation:

- Internal data sources
- Extension - Standard
- Extension - Catalog

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `copy`*`(request)`*

Faults

Refer to **Error Handling on page 164** for a description of any faults that may be returned.

The API does not allow the following scenarios and throws an `InvalidObjectFault`:

- No `dataSourceRecord` specified in the request.
- More than one `dataSourceRecord` specified in the request.
- A non-existent column in specified in the input record.
- A system column in specified in the input record.
- A column is specified more than one in the input record.

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

public void upsertRecord(InternalDataSourceId dataSourceId) throws Exception
{
  addSampleRecordForUpsert(dataSourceId);

/* Wait for data source status to be idle before adding */
  helper.waitForIdle(dataSourceId);

/* A non-existent record, which will get added. */
  UpsertDataSourceRecordsRequest upsertRequest1 = new UpsertDataSourceRecordsRequest();
  upsertRequest1.setDataSourceId(dataSourceId);
  DataSourceRecord newRecord = new DataSourceRecord();
  NameValuePair idCol = new NameValuePair();
  idCol.setName("id");
  idCol.setValue("903");
  newRecord.getField().add(idCol);
  NameValuePair emailCol = new NameValuePair();
  emailCol.setName("email_address");
  emailCol.setValue("email_for_cxf_upsert3@cxf.com");
  newRecord.getField().add(emailCol);
  NameValuePair nameCol = new NameValuePair();
  nameCol.setName("name");
  nameCol.setValue("cxf_name_for_upsert3");
  newRecord.getField().add(nameCol);
  upsertRequest1.getDataSourceRecord().add(newRecord);
  service.upsertRecord(upsertRequest1);

/* Wait for data source status to be idle before adding */
  helper.waitForIdle(dataSourceId);

/* An existing record, which will get updated. */
  UpsertDataSourceRecordsRequest upsertRequest2 = new UpsertDataSourceRecordsRequest();
  upsertRequest2.setDataSourceId(dataSourceId);
  DataSourceRecord updateRecord = new DataSourceRecord();
  idCol = new NameValuePair();
  idCol.setName("id");
  idCol.setValue("901");
  updateRecord.getField().add(idCol);
  emailCol = new NameValuePair();
  emailCol.setName("email_address");
  emailCol.setValue("email_for_cxf_upsert-updat- ed@cxf.com");
  updateRecord.getField().add(emailCol);
  upsertRequest2.getDataSourceRecord().add(updateRecord);
  service.upsertRecord(upsertRequest2);
}
```

Request Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/

<xs:complexType name="UpsertDataSourceRecordsRequest">
  <xs:complexContent>
    <xs:extension base="tns:UpsertRecordsRequest">
      <xs:sequence>
        <xs:element name="dataSourceId" type="tns- :DataSourceId"/>
        <xs:element name="dataSourceRecord" type="tns- :DataSourceRecord"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Response Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                    *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                            *
*                                                                    *
* ---------------------------------------------------------------- *
*/
<xs:complexType name="UpsertRecordsResponse">
  <xs:complexContent>
    <xs:extension base="tns:Response">
      <xs:sequence>
        <xs:element name="success" type="xs:boolean"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

`validateXsl()`

Use the `validateXsl()` operation to validate the XSL in an XSL template.

Supported Assets

The following list shows the Message Studio assets that are supported for the `validateXsl()` operation:

- Template

Recommendations

None.

Code Sequence

1. `list`*`(filter)`*
2. `validateXsl`*`(request)`*

Supported Requests

The `validateXsl()` operation supports a single object that can be passed in as an argument. The object is associated with a functional area of the Message Studio UI, and is also mapped to a specific SOAP request.

The following table maps the functional area to the object/request.

Functional area	Object/Request
Template	`ValidateXslRequest (TemplateId)`

Response

    validateXslResponse

Faults

Refer to **Error Handling on page 164** for a description of the following faults that may be returned:

- `objectNotFoundFault`
- `authorizationFault`
- `unexpectedFault`

Sample Code

```
/*
* ----------------------------------------------------------------- *
*  STRONGVIEW SYSTEMS                                               *
*                                                                   *
*  Copyright © 2014 StrongView Systems, Inc. - All rights reserved. *
*                                                                   *
*  Visit http://www.strongview.com for more information             *
*                                                                   *
*  You may incorporate this Source Code in your application only    *
*  you own a valid license to do so from StrongView Systems, Inc    *
*  and the copyright notices are not removed from the source cod    *
*                                                                   *
*  Distributing our source code outside your organization           *
*  is strictly prohibited                                           *
*                                                                   *
* ----------------------------------------------------------------- *
*/
/**
* Validate if a template's xsl content is valid.
*
* @param templateId the TemplateId whose XSL content will be validated.
* @return if the content is valid XSL
* @throws Exception any exception
*/
```

```
public boolean validateXsl(TemplateId templateId) throws Exception {
  ValidateXslRequest request = new ValidateXslRequest();
  request.setMessageFormat(MessageFormat.HTML);
  request.setTemplateId(templateId);
  return service.validateXsl(request).isValid();
}
```

Error Handling

When you pass SOAP calls using the API, Message Studio may return one of the faults below:

Fault	**Definition**
`authorizationFault`	Occurs when log into Message Studio (through the SOAP header) with a user that does not have the appropriate permissions to execute the operation.
`concurrentModificationFault`	Occurs when a different user modifies an asset between the time you perform the `get()` and the `update()`.
`objectNotFoundFault`	Occurs when you attempt to access an asset that does not exist within Message Studio. For example, if you call an operation on an `objectId` that doesn't exist within Message Studio.
`invalidObjectFault`	Occurs when an object that was passed into the operation includes a parameter with an invalid value.
`staleObjectFault`	Occurs when the server has a new copy than the one held by the API. You must re-get the same object before performing any operation.
`unexpectedFault`	Occurs when an error, other than the ones listed here, occurs.
`unrecognizedObjectTypeFault`	Occurs when you pass the wrong kind of object to the operation. For example, if you tried to perform a `refreshRecords()` operation on a mailing instead of on an external data source.

Resources

The Message Studio web service is organized as a set of objects defined in the XSD file, and a set of operations defined in the WSDL file. This new API uses the same authentication and authorization model as Message Studio.

HTML Description

Once you have installed Message Studio, you can get an HTML description of the service file by using:

```
https://hostname:4443/sm/services/
```

where:

hostname is the hostname of your Message Studio server.

This page also includes the service endpoint URL.

WSDL

The WSDL is available from the following location for each version of the API:

```
https://hostname:4443/sm/services/mailing/version?wsdl
```

where:

- *hostname*—Hostname of your Message Studio server.
- *version*—Version of the Web Services API (e.g., `2009/03/02, v2`)

XSD

The XSD is available from the following location for each version of the API:

```
https://hostname:4443/s-
m/services/mailing/version?xsd=MailingServiceSchema.xsd
```

where:

- *hostname* is the hostname of your Message Studio server.

- *version*—Version of the Web Services API (e.g., `2009/03/02, v2`)

All required parameters for each operation are defined in the XSD. Optional parameters will have `minOccurs=”0”`. Any other value for `minOccurs` indicates a required parameter. If `minOccurs` is not specified, it defaults to 1 and is required.

Action Status

When you call an action, you must wait for a specific status to make sure the action has suc- cessfully completed before calling another action for the same mailing. The following table indi- cates the actions you can call and the status you must wait for before calling a second action:

Action	Wait for
Pause	PAUSED
Load	ACTIVE
Cancel	CANCELED
Resume	ACTIVE
Launch	ACTIVE
Close	COMPLETED

Mailing status is defined in the MailingServiceSchema.xsd, MailingStatus type.

Network Considerations

By default, the Message Studio web service runs on port 4443. This is configured during Message Studio installation.

Sample Code

Sample code is available on the Message Studio server in the following directory:

```
/data1/message_studio/lib/sm/webservice_samples.tar.gz
```

In addition, you can access the code from a browser:

```
https://hostname/sm/webservice_samples.tar.gz
```

where:

hostname—Hostname of your Message Studio server