SMTP Transport Reference

SMTP Transport Reference

Your Rating: ☆☆☆☆☆ Results: ★★★★★ 0 rates

Introduction

The SMTP transport can be used for sending messages over SMTP using the `javax.mail` API. The implementation supports CC/BCC/ReplyTo addresses, attachments and custom Header properties. It also provides support for `javax.mail.Message` transformation. The SMTPS connector enables SMTP over SSL using the `javax.mail` APIs. It supports all the elements and attributes of the SMTP transport, plus some required properties for setting up the client key store and the trust store for the SSL connection.

TLS/SSL connections are made on behalf of an entity, which can be anonymous or identified by a certificate. The *key store* provides the certificates and associated private keys necessary for identifying the entity making the connection. Additionally, connections are made to trusted systems. The public certificates of trusted systems are stored in a *trust store*, which is used to verify that the connection made to a remote system matches the expected identity.

Transport Info

Transport	Doc	Inbound	Outbound	Request	Transactions	Streaming	Retries	MEPs	Default MEP	Maven Artifact
SMTP	JavaDoc SchemaDoc	✗	✓	✓	✗	✗	✗	one-way	one-way	org.mule.transport:m
SMTPS	JavaDoc SchemaDoc	✗	✓	✓	✗	✗	✗	one-way	one-way	org.mule.transport:m

Legend

Click here to expand...

Transport - The name/protocol of the transport
Docs - Links to the JavaDoc and SchemaDoc for the transport
Inbound - Whether the transport can receive inbound events and can be used for an inbound endpoint
Outbound - Whether the transport can produce outbound events and be used with an outbound endpoint
Request - Whether this endpoint can be queried directly with a request call (via MuleClinet or the EventContext)
Transactions - Whether transactions are supported by the transport. Transports that support transactions can be configured in either local or distributed two-phase commit (XA) transaction.
Streaming - Whether this transport can process messages that come in on an input stream. This allows for very efficient processing of large data. For more information, see Streaming.
Retry - Whether this transport supports retry policies. Note that all transports can be configured with Retry policies, but only the ones marked here are officially supported by MuleSoft
MEPs - Message Exchange Patterns supported by this transport
Default MEP - The default MEP for endpoints that use this transport that do not explicitly configure a MEP
Maven Artifact - The group name a artifact name for this transport in Maven

Namespace and Syntax

XML namespace:

```
xmlns:smtp "http://www.mulesoft.org/schema/mule/smtp"
xmlns:smtps "http://www.mulesoft.org/schema/mule/smtps"
```

XML Schema location:

```
http://www.mulesoft.org/schema/mule/smtp http://www.mulesoft.org/schema/mule/smtp/3.1/mule-smtp.xsd
http://www.mulesoft.org/schema/mule/smtps http://www.mulesoft.org/schema/mule/smtps/3.1/mule-smtps.xsd
```

Connector syntax:

```
<smtp:connector name="smtpConnector" bccAddresses="abc@example.com" ccAddresses="bcd@example.com"
contentType="foo/bar"
fromAddress="cde@example.com" replyToAddresses="def@example.com"
subject="subject">
  <smtp:header key="foo" value="bar" />
  <smtp:header key="baz" value="boz" />
</smtp:connector>

<smtps:connector name="smtpsConnector" bccAddresses="abc@example.com" ccAddresses="bcd@example.com"
contentType="foo/bar"
fromAddress="cde@example.com" replyToAddresses="def@example.com"
subject="subject">
  <smtps:header key="foo" value="bar" />
  <smtps:header key="baz" value="boz" />
  <smtps:tls-client path="clientKeystore" storePassword="mulepassword" />
  <smtps:tls-trust-store path="greenmail-truststore" storePassword="password" />
</smtps:connector>

<smtp:gmail-connector name="smtpGmailConnector"  bccAddresses="abc@example.com" ccAddresses=
"bcd@example.com" contentType="foo/bar"
fromAddress="cde@example.com" replyToAddresses="def@example.com"
subject="subject">
  <smtp:header key="foo" value="bar" />
  <smtp:header key="baz" value="boz" />
</smtp:gmail-connector>
```

Endpoint syntax:
You can define your endpoints 2 different ways:

1. Prefixed endpoint:

```
<smtp:outbound-endpoint host="localhost" port="65437" from="steve@mycompany.com"
                        to="bob@example.com" subject="Please verify your account details"/>
<smtps:outbound-endpoint host="localhost" port="65437" from="steve@mycompany.com"
                        to="bob@example.com" subject="Please verify your account details"/>
```

2. Non-prefixed URI:

```
smtp://muletestbox:123456@smtp.mail.yahoo.co.uk?address=dave@mycompany.com
smtps://muletestbox:123456@smtp.mail.yahoo.co.uk?address=dave@mycompany.com
```

See the sections below for more information.

Features

- Simple to configure email access on outbound endpoints
- Easy to configure tls security

Usage

If you want to include the SMTP email transport in your configuration, these are the namespaces you need to define:

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:spring="http://www.springframework.org/schema/beans"
xmlns:smtp="http://www.mulesoft.org/schema/mule/smtp"
xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/3.1/mule.xsd
http://www.mulesoft.org/schema/mule/smtp http://www.mulesoft.org/schema/mule/smtp/3.1/mule-smtp.xsd">
...
```

Secure version:

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:spring="http://www.springframework.org/schema/beans"
xmlns:smtps="http://www.mulesoft.org/schema/mule/smtps"
xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/3.1/mule.xsd
http://www.mulesoft.org/schema/mule/smtps
http://www.mulesoft.org/schema/mule/smtps/3.1/mule-smtps.xsd">
```

Then you need to configure your connector and endpoints as described below.

Configuration Example

Say your CFO wants an email notification of all processed orders. The following configuration will pick up any files in the 'processed' directory, convert them to a string and send it as the email body to the CFO.

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:spring="http://www.springframework.org/schema/beans"
xmlns:smtps="http://www.mulesoft.org/schema/mule/smtps"
xmlns:vm="http://www.mulesoft.org/schema/mule/vm"
xmlns:file="http://www.mulesoft.org/schema/mule/file"
xmlns:email="http://www.mulesoft.org/schema/mule/email"
xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/3.1/mule.xsd
http://www.mulesoft.org/schema/mule/file http://www.mulesoft.org/schema/mule/file/3.1/mule-file.xsd
http://www.mulesoft.org/schema/mule/smtps http://www.mulesoft.org/schema/mule/smtps/3.1/mule-smtps.xsd
http://www.mulesoft.org/schema/mule/email http://www.mulesoft.org/schema/mule/email/3.1/mule-email.xsd
http://www.mulesoft.org/schema/mule/vm http://www.mulesoft.org/schema/mule/vm/3.1/mule-vm.xsd">

    <smtp:connector name="smtpConnector"  />

    <flow name="processed-orders">
        <file:inbound-endpoint path="/tmp/processed">
            <file:file-to-string-transformer/>
        </file:inbound-endpoint>
        <smtp:outbound-endpoint host="smtpsServer" port="25" from="bob" subject="processed order" to=
"cfo@example.com">
            <email:string-to-email-transformer/>
        </smtps:outbound-endpoint>
    </flow>
</mule>
```

This configuration defines a inbound file endpoint which looks in the '/tmp/processed' directory and converts any files found to a string . An outbound smtp server is defined on . A string-to-email-transformer will convert the string to email format before the email is sent.

Secure version:

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:spring="http://www.springframework.org/schema/beans"
xmlns:smtps="http://www.mulesoft.org/schema/mule/smtps"
xmlns:vm="http://www.mulesoft.org/schema/mule/vm"
xmlns:file="http://www.mulesoft.org/schema/mule/file"
xmlns:email="http://www.mulesoft.org/schema/mule/email"
xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/3.1/mule.xsd
http://www.mulesoft.org/schema/mule/file http://www.mulesoft.org/schema/mule/file/3.1/mule-file.xsd
http://www.mulesoft.org/schema/mule/smtps http://www.mulesoft.org/schema/mule/smtps/3.1/mule-smtps.xsd
http://www.mulesoft.org/schema/mule/email http://www.mulesoft.org/schema/mule/email/3.1/mule-email.xsd
http://www.mulesoft.org/schema/mule/vm http://www.mulesoft.org/schema/mule/vm/3.1/mule-vm.xsd">

    <smtps:connector name="smtpsConnector">
        <smtps:tls-client path="clientKeystore" storePassword="mulepassword" />
        <smtps:tls-trust-store path="greenmail-truststore" storePassword="password" />
    </smtps:connector>

    <flow name="processed-orders">
        <file:inbound-endpoint path="/tmp/processed">
            <file:file-to-string-transformer/>
        </file:inbound-endpoint>
        <smtps:outbound-endpoint host="smtpsServer" port="25" from="bob" subject="processed order" to=
"cfo@example.com">
            <email:string-to-email-transformer/>
        </smtps:outbound-endpoint>
    </flow>
</mule>
```

The smtps connector has tls client and server keystore information as defined on . An inbound file endpoint looks in the '/tmp/processed' directory and converts any files found to a string . An outbound smtp server is defined on . A string-to-email-transformer will convert the string to email format before the email is sent.

Configuration Reference

Connectors

The SMTP connector supports all the common connector attributes and properties and the following optional elements and attributes:

Attribute	Description	Default	Required
bccAddresses	Comma separated list of addresses for blind copies.		False
ccAddresses	Comma separated list of addresses for copies.		False
contentType	Mime type for the outgoing message.		False
fromAddress	The from address for the outgoing message.		False
replyToAddresses	The reply-to address for the outgoing message.		False
subject	The default subject for the outgoing message if none is set in the message.		False

Element	Description
header	Additional header name and value, added to the message.

For the secure version, the following elements are also required:

Element	Description

tls-client	Configures the client key store with the following attributes: • path: The location (which will be resolved relative to the current classpath and file system, if possible) of the keystore that contains public certificates and private keys for identification • storePassword: The password used to protect the keystore • class: The type of keystore used (a Java class name)
tls-trust-store	Configures the trust store. The attributes are: • path: The location (which will be resolved relative to the current classpath and file system, if possible) of the trust store that contains public certificates of trusted servers • storePassword: The password used to protect the trust store

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:spring="http://www.springframework.org/schema/beans"
       xmlns:smtp="http://www.mulesoft.org/schema/mule/smtp"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
       http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/3.0/mule.xsd
       http://www.mulesoft.org/schema/mule/smtp
http://www.mulesoft.org/schema/mule/smtp/3.0/mule-smtp.xsd">
...
<smtp:connector name="smtpConnector" bccAddresses="abc@example.com" ccAddresses="bcd@example.com"
contentType="foo/bar"
fromAddress="cde@example.com" replyToAddresses="def@example.com"
subject="subject">
  <smtp:header key="foo" value="bar" />
  <smtp:header key="baz" value="boz" />
</smtp:connector>
```

Secure version:

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:spring="http://www.springframework.org/schema/beans"
       xmlns:smtps="http://www.mulesoft.org/schema/mule/smtps"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
       http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/3.0/mule.xsd
       http://www.mulesoft.org/schema/mule/smtps
http://www.mulesoft.org/schema/mule/smtps/3.0/mule-smtps.xsd">

<smtps:connector name="smtpsConnector">
  <smtps:tls-client path="clientKeystore" storePassword="mulepassword" />
  <smtps:tls-trust-store path="greenmail-truststore" storePassword="password" />
</smtps:connector>
<model name="test">
  <service name="relay">
    <inbound>
      <vm:inbound-endpoint path="send" />
  </inbound>
  <outbound>
    <pass-through-router>
      <smtps:outbound-endpoint host="localhost" port="65439" to="bob@example.com" />
    </pass-through-router>
  </outbound>
...
```

The gmail-connector connector supports all of the above.
For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:spring="http://www.springframework.org/schema/beans"
       xmlns:smtp="http://www.mulesoft.org/schema/mule/smtp"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
       http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/3.0/mule.xsd
       http://www.mulesoft.org/schema/mule/smtp
http://www.mulesoft.org/schema/mule/smtp/3.0/mule-smtp.xsd">
...
<smtp:gmail-connector name="smtpGmailConnector"  bccAddresses="abc@example.com" ccAddresses=
"bcd@example.com" contentType="foo/bar"
fromAddress="cde@example.com" replyToAddresses="def@example.com"
subject="subject">
  <smtp:header key="foo" value="bar" />
  <smtp:header key="baz" value="boz" />
</smtp:gmail-connector>
```

Endpoints

SMTP endpoints describe details about the SMTP server and the recipients of messages sent from the SMTP endpoint. You configure the endpoints just as you would with any other transport, with the following additional attributes:

Attribute	Description
user	The user name of the mailbox owner
password	The password of the user
host	The IP address of the SMTP server, such as www.mulesoft.com, localhost, or 127.0.0.1

port	The port number of the SMTP server
to	The destination for the email
from	The address of the sender of the email
subject	The email subject
cc	A comma-separated list of email addresses to copy on this email
bcc	A comma-separated list of email addresses to blind-copy on this email
replyTo	The address used by default if someone replies to the email

For example:

```
<outbound>
  <pass-through-router>
    <smtp:outbound-endpoint host="localhost" port="65437" from="steve@mycompany.com"
                            to="bob@example.com" subject="Please verify your account details"/>
  </pass-through-router>
</outbound>
```

Secure version:

```
<outbound>
  <pass-through-router>
    <smtps:outbound-endpoint host="localhost" port="65437" from="steve@mycompany.com"
                            to="bob@example.com" subject="Please verify your account details"/>
  </pass-through-router>
</outbound>
```

You can also define the endpoints using a URI syntax:

```
<inbound-endpoint address="smtp://muletestbox:123456@smtp.mail.yahoo.co.uk?address=dave@mycompany.com"
/>
<inbound-endpoint address="smtps:
//muletestbox:123456@smtp.mail.yahoo.co.uk?address=dave@mycompany.com"/>
```

This will send mail using `smtp.mail.yahoo.co.uk` (using the default SMTP port) to the address `dave@mycompany.com`. The SMTP request is authenticated using the username `muletestbox` and the password `123456`.

Transformers

These are transformers specific to this transport. Note that these are added automatically to the Mule registry at start up. When doing automatic transformations these will be included when searching for the correct transformers.

Name	Description
email-to-string-transformer	Converts an email message to string format.
string-to-email-transformer	Converts a string message to email format.
object-to-mime-transformer	Converts an object to MIME format.
mime-to-bytes-transformer	Converts a MIME message to a byte array.

bytes-to-mime-transformer	Converts a byte array message to MIME format.

Here is how you define transformers in your Mule configuration file:

```
<email:bytes-to-mime-transformer encoding="" ignoreBadInput="" mimeType="" name="" returnClass=""
xsi:type=""/>
<email:email-to-string-transformer encoding="" ignoreBadInput="" mimeType="" name="" returnClass=""
xsi:type=""/>
<email:mime-to-bytes-transformer encoding="" ignoreBadInput="" mimeType="" name="" returnClass=""
xsi:type=""/>
<email:object-to-mime-transformer encoding="" ignoreBadInput="" mimeType="" name="" returnClass=""
useInboundAttachments="true" useOutboundAttachments="true"/>
{Note}Need to explain attributes somewhere; can we pull them in from xsd?{Note}
<email:string-to-email-transformer encoding="" ignoreBadInput="" mimeType="" name="" returnClass=""
xsi:type=""/>
```

Each transformer supports all the common transformer attributes and properties:

Transformer

A reference to a transformer defined elsewhere.

Attributes of <transformer...>

Name	Type	Required	Default	Description
name	name (no spaces)	no		Identifies the transformer so that other elements can reference it. Required if the transformer is defined at the global level.
returnClass	string	no		The class of the message generated by the transformer. This is used if transformers are auto-selected and to validate that the transformer returns the correct type. Note that if you need to specify an array type you need postfix the class name with '[]'. For example, if you want return a an Orange[], you set the return class to 'org.mule.tck.testmodels.fruit.Orange[]'.
ignoreBadInput	boolean	no		Many transformers only accept certain classes. Such transformers are never called with inappropriate input (whatever the value of this attribute). If a transformer forms part of a chain and cannot accept the current message class, this flag controls whether the remaining part of the chain is evaluated. If true, the next transformer is called. If false the chain ends, keeping the result generated up to that point.
encoding	string	no		String encoding used for transformer output.
mimeType	string	no		The mime type, e.g. text/plain or application/json
ref	string	yes		The name of the transformer to use.

Child Elements of <transformer...>

Name	Cardinality	Description

The object-to-mime-transformer has the following attributes:

Attribute	Description	Default Value
useInboundAttachments	Whether to transform inbound attachment in the input message into MIME parts.	true
useOutboundAttachments	Whether to transform outbound attachment in the input message into MIME parts.	true

To use these transformers, make sure you include the 'email' namespace in your mule configuration.

Filters

Filters can be set on an endpoint to filter out any unwanted messages. The Email transport provides a couple of filters that can either be used directly or extended to implement custom filtering rules.

Filter	Description
org.mule.providers.email.filters.AbstractMailFilter	A base filter implementation that must be extended by any other mail filter.
org.mule.providers.email.filters.MailSubjectRegExFilter	Applies a regular expression to a Mail Message subject.

This is how you define the MailSubjectRegExFilter in your Mule configuration:

```
<message-property-filter pattern="to=barney@mule.org"/>
```

The 'pattern' attribute is a regular expression pattern. This is defined as java.util.regex.Pattern.

Exchange patterns / features of the transport

(see transport matrix)

Schema Reference

You can view the full schema for the SMTP email transport here . The secure version is here .

Java API Reference

The Javadoc for this transport can be found here .

Maven module

The email transports are implemented by the mule-transport-email module. You can find the source for the email transport under transports/email.

If you are using maven to build your application, use the following dependency snippet to include the email transport in your project:

```
<dependency>
  <groupId>org.mule.transports</groupId>
  <artifactId>mule-transport-email</artifactId>
</dependency>
```

If you are building Mule ESB from source or including mule artifacts in your maven project, it may be necessary to add the 'mule-deps' repository to your maven configuration. This repository contains 3rd party binaries which may not be in any other public maven repository.

To add the 'mule-deps' repository to your maven project, add the following to your pom.xml:

```
<repositories>
    <repository>
        <id>mule-deps</id>
        <name>Mule Dependencies</name>
        <url>http://dist.codehaus.org/mule/dependencies/maven2</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </repository>
</repositories>
```

Limitations

The following known limitations affect email transports:

- Retry policies do not work with email transports
- Timeouts are not supported in email transports
- Can't send same object to different email users
- MailSubjectRegExFilter cannot handle mails with attachments

So far, all configuration has been static, in that you define all the information in the configuration of the endpoint. However, you can set the connector properties to control the settings of the outgoing message. These properties will override the endpoint properties. If you always want to set the email address dynamically, you can leave out the `to` attribute (or the `address` parameter if you're using URIs} on the SMTP endpoint.

Escape Your Credentials
If you use a URI-style endpoint and you include the user name and password, escape any characters that are illegal for URIs, such as the @ character. For example, if the user name is `user@myco.com`, you should enter it as `user%40myco.com`.