EUROPEAN

MIDDLEWARE INITIATIVE

APEL SSM USER GUIDE

Document Version:	**1.1**
EMI Component Version:	**2.0.0**
Date:	**23.07.2013**

This work is co-funded by the European Commission as part of the EMI project under Grant Agreement INFSO-RI-261611.

DOCUMENT CHANGE LOG

Version	Date	Comment	Author
1.1	23/07/2013	Added detail about use_ssl and destination queue.	Stuart Pullinger
1.0	12/02/2013	Initial version	Will Rogers

Contents

1. INTRODUCTION

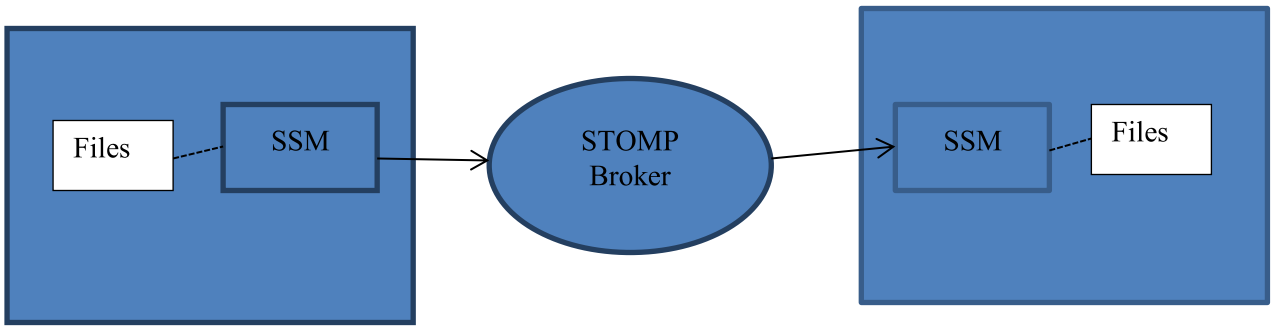
SSM (Secure STOMP Messenger) is a simple program which will transfer files between computers using the Simple Text-Oriented Messaging Protocol (STOMP). Messages are signed and may be encrypted during transit, to ensure integrity and security.

SSM acts both as a sender and a receiver. However, only the sending capability is supported in EMI.

2. ARCHITECTURE

SSM is based around files. It will read files from a local filesystem and send them as messages to a specified queue on a broker. Each file will be sent as one message. A receiving SSM, listening to that queue, will verify the messages and write them to its local filesystem.

SSM uses the python-dirq library to manage interaction with the filesystem. OpenSSL is used directly for cryptography.



3. DEPLOYMENT

Detailed installation instructions for SSM can be found in the APEL SSM System Administrator Guide.

SSM will typically be deployed in one of two ways:

- As a stand-alone program, perhaps to send StAR messages to the APEL server
- As part of the APEL Publisher

Both deployments will read from the standard configuration file /etc/apel/sender.cfg. The APEL client package will automatically produce files which SSM can find, but if using SSM as a stand-alone product you must add messages correctly.

3.1.1 Stand-alone SSM

For installing SSM, see the APEL SSM System Administrator Guide.

To add messages for sending by SSM, see section 4 of this document.

3.1.2 SSM as part of APEL Publisher

For installing SSM as part of APEL Publisher, see the APEL Publisher System Administrator Guide.

Messages will automatically be sent using the APEL Publisher program, but you may still change configuration options as described below.

3.2. CONFIGURATION

The APEL SSM configuration file is located at `/etc/apel/sender.cfg`.

Each section of the configuration file is described below.

3.2.1 broker

This section allows you to configure the connection to the broker.

```
[broker]

# The SSM will query a BDII to find brokers available.  These details
# are for the EGI production broker network
bdii: ldap://lcg-bdii.cern.ch:2170
network: PROD
# OR (these details will only be used if the broker network
# settings aren't used)
#host: test-msg01.afroditi.hellasgrid.gr
#port: 6163

# broker authentication.  If use_ssl is set, the certificates configured
# in the mandatory [certificates] section will be used.
use_ssl: true
```

You may specify a broker in two ways.

If you specify a BDII and a broker network name, SSM will query the BDII for available brokers in that network and choose one of those brokers. The valid networks are 'PROD' for the production network and 'TEST-NWOB' for the test network.

Alternatively, you may choose a broker by specifying its hostname and port.

You should connect using SSL to the production network ('PROD') by setting

```
use_ssl: true
```

You should connect without using SSL to the test network ('TEST-NWOB') by setting:

```
use_ssl: false
```

If you are specifying host and port, make sure that you choose the correct port for either STOMP or STOMP+SSL.

3.2.2 certificates

This section contains information about the certificates used for cryptography.

```
[certificates]
certificate: /etc/grid-security/hostcert.pem
key: /etc/grid-security/hostkey.pem
# If this is supplied, outgoing messages will be encrypted
# using this certificate
#server: /etc/grid-security/servercert.pem
capath: /etc/grid-security/certificates
```

SSM uses X509 certificates to sign all messages, so that the receiver can verify the integrity of the message. SSM uses OpenSSL for cryptography and all certificates must be in PEM format. SSM verifies certificates against the CA certificates in the capath directory.

Using SSL connections is sufficient to protect the contents of the message. However, you may optionally encrypt messages by using the `server` option to specify the location of the receiving server's certificate. You will need to request the receiving server certificate from the APEL team. Please raise a GGUS ticket with this request.

3.2.3 messaging

This section contains information about the messages.

```
[messaging]

# Topic that we are to send messages to
destination:

# Outgoing messages will be read and removed from this directory.
path: /var/spool/apel/outgoing
```

The destination queue on the broker, to which messages will be sent, is configured here. For production services the destination queue for CPU accounting is:

```
destination: /queue/global.accounting.cpu.central
```

For test services the destination queue for CPU accounting is:

```
destination: /queue/global.accounting.cputest.CENTRAL
```

(A test service is provided for cloud and storage accounting records. The destination queues for these are different. For cloud:

```
destination: /queue/global.accounting.cloudtest.CENTRAL
```

For storage:

```
destination: /queue/global.accounting.storagetest.CENTRAL
```
).

The path option allows you to specify a filesystem location from which messages will be read.

3.2.4 logging

```
[logging]
logfile = /var/log/apel/sender.log
level = INFO
console = true
```

This controls the logging options:

- logfile: specifies the location of the log file to write
- level: can be DEBUG, INFO, WARNING, ERROR, CRITICAL. It is recommended to leave this as INFO by default.
- console: whether to log to the console as well as to the log file.

3.3. LOGGING

The default log file for the sending SSM is `/var/log/apel/sender.log`.

4. ADDING MESSAGES

There are two ways of adding messages to SSM:

- Programmatically, using perl or python
- Manually, writing files to the filesystem

Each file will be sent as one message. We recommend that messages are no bigger than 1MB.

4.1. PROGRAMMATICALLY

You can add messages to SSM using the `python-dirq`[1] or the `perl-Directory-Queue`[2] libraries.

Create a QueueSimple object with path `/var/spool/apel/outgoing` and add messages.

4.2. MANUALLY

If you are not using perl or python, you can simply write files to the filesystem and SSM will send them. However, there are constraints on the filenames:

- Files must be in the location `/var/spool/apel/outgoing/<directory>/<file>`
- **<directory> must be 8 hex characters** - that is 0-9 or a-f
- **<file> must be 14 hex characters**

[1] http://pypi.python.org/pypi/dirq
[2] http://search.cpan.org/dist/Directory-Queue/

4.2.1 Example naming scheme

These notes are taken from the dirq documentation and describe a possible (not compulsory) naming scheme.

```
Directory Structure
```

```
The top-level directory contains intermediate directories that contain the
stored elements, each of them in a file.

The names of the intermediate directories are time based: the element
insertion time is used to create an 8-digits long hexadecimal number. The
granularity (see the constructor) is used to limit the number of new
directories. For instance, with a granularity of 60 (the default), new
directories will be created at most once per minute.

Since there is usually a filesystem limit in the number of directories a
directory can hold, there is a trade-off to be made. If you want to support
many added elements per second, you should use a low granularity to keep
small directories. However, in this case, you will create many directories
and this will limit the total number of elements you can store.

The elements themselves are stored in files (one per element) with a 14-
digits long hexadecimal name SSSSSSSSMMMMMR where:
```

- `SSSSSSSS represents the number of seconds since the Epoch`
- `MMMMM represents the microsecond part of the time since the Epoch`
- `R is a random digit used to reduce name collisions`

5. SENDING MESSAGES

If you are not running the APEL Publisher to send messages, run:

```
# /usr/bin/ssmsend
```