

## Using EMG 3

This document describes how to use new major functionality in EMG 3. In addition to the functionality described below there are numerous minor features and new configuration keywords available. These will be described in the final EMG 3 User's Guide when EMG is officially released.

### ***Connector templates***

When defining many connectors that have similar properties it can be useful to first define a connector template and then let other connectors inherit from the template. A connector can inherit from any connector, but if the keyword VIRTUAL is used for the template connector it cannot be used itself and it does not need to contain a complete configuration.

Sample connector configuration:

```
# HTTP connector template
CONNECTOR http-tmpl <
TYPE=OUTGOING
PROTOCOL=HTTP
INSTANCES=1
UDHVIAOPTIONAL
MAPPING=mappings/
# This is not a real connector
VIRTUAL
>

CONNECTOR http-client1 <
INHERIT=http-tmpl
ADDRESS=http://www.example.com/client1
>

CONNECTOR http-client2 <
INHERIT=http-tmpl
ADDRESS=http://www.example.com/client2
>
```

### ***Source Address Translation (SAT)***

When messages are received from clients that do not have a unique MSISDN number such as e-mail clients some kind of address translation may be needed before sending the messages to mobile phones. Also when replies to previously sent SMS are received from a phone there is no way of knowing exactly which original message the response relates to unless some more elaborate methods are used.

EMG SAT solves both these problems by making it possible to allocate sender addresses from a pool in an intelligent way. As long as the combination of source address and destination address is chosen in a way that it is unique it is possible to map a non-MSISDN sender to a valid MSISDN requiring a relatively small pool of addresses (one address is needed per sender for a particular destination, 10 could be enough).

In order to be able to distinguish exactly which message a recipient answers to (THREADED) a larger (but still not very large) pool is needed (one addresses is needed for the maximum number of messages any destination may receive, 50-100 could be enough).

EMG uses different keywords for specifying where a translation entry should be created and when it should be looked up before transmission. This can be performed when messages are received (SATPOOL\_CREATE\_IN, SATPOOL\_LOOKUP\_IN) or as they are sent out (SATPOOL\_CREATE, SATPOOL\_LOOKUP).

For lookups multiple SAT pools can be specified by using the SATPOOL\_LOOKUP keywords repeatedly.

Sample connector configuration:

```
SATPOOL sat1 <
# We allocate a pool with 10 addresses
ADDRESSRANGE=467012340-467012349
THREADED
>

SATPOOL sat2 <
# We allocate another pool with 10 addresses
ADDRESSRANGE=467012350-467012359
THREADED
>

CONNECTOR smtp-in1 <
TYPE=INCOMING
ADDRESS=192.168.0.1:25
PROTOCOL=SMTP
INSTANCES=10
SATPOOL_CREATE_IN=sat1
ROUTE=smc
>

CONNECTOR smtp-in2 <
TYPE=INCOMING
ADDRESS=192.168.0.1:25
PROTOCOL=SMTP
INSTANCES=10
SATPOOL_CREATE_IN=sat2
ROUTE=smc
>

CONNECTOR smtp-out <
TYPE=OUTGOING
ADDRESS=#MX
PROTOCOL=SMTP
INSTANCES=1
```

>

```
CONNECTOR smsc <
TYPE=OUTGOING
PROTOCOL=SMPP
INSTANCES=1
ROUTE=smtp-out
SATPOOL_LOOKUP_IN=sat1
SATPOOL_LOOKUP_IN=sat2
>
```

### ***Regular expressions***

Now Perl regular expressions can be used to rewrite addresses as well as message content. Related connector keywords: REGEXP\_DESTADDR, REGEXP\_DESTADDR\_IN, REGEXP\_MESSAGE, REGEXP\_SOURCEADDR, REGEXP\_SOURCEADDR\_IN.

Sample syntax:

```
REGEXP=s/and/or/g
```

The above would replace all occurrences of “and” with “or” in the message body. This option can be given several times for a single connector. The substitutions will be performed in the order they are given.

This functionality is enabled via PCRS, written and copyright (C) 2000, 2001 by Andreas S. Oesterhelt, andreas@oesterhelt.org. For more information about capabilities please visit <http://www.oesterhelt.org/pcrs>.

### ***Retry schemes***

In EMG 3 it is possible to define customized retry schemes for all protocols. A retry scheme determines how different protocol-specific error codes should be handled by EMG when connecting, logging in and sending and receiving messages. For message-related protocol errors it can be determined how many retries should be done, if any, by which interval before a message should be considered permanently failed.

```
CONNECTOR smpp-out <
TYPE=OUTGOING
ADDRESS=10.0.0.1:5000
PROTOCOL=SMPP
INSTANCES=1
RETRYScheme=/etc/emg/retryschemes/smpp
>
```

Format of retry scheme file is one entry per row each entry using 3 to 7 columns.

<b>Column</b>	<b>Description</b>
1	C or M for Connector or Message entry respectively
2	Error code. * (any error code), a number or a range can be used

3	Retry time, in seconds
4	Connects
5	Max sleep, in seconds
6	Hold delay, in seconds
7	Flags

### ***Extended SMTP support***

EMG can now act as an SMTP server being able of delivering messages based on MX lookups. Multiple deliveries can be performed in parallel using different threads. However, EMG is intelligent and does not try to use multiple threads towards the same SMTP server.

Sections in the configuration file where attributes per domain are given can be specified.

SESSIONS	Max number of simultaneous sessions
MAILSPERSESSION	Max number of e-mails to send per SMTP session
MAILSPERMINUTE	Max number of e-mail per minute to send
RETRYTIME	Time to wait before retry in case of error
PORT	Specifies another port than port 25 should be used

Sample connector configuration:

```
DOMAIN hotmail.com <
SESSIONS=1
MAILSPERMINUTE=3
>
```

```
CONNECTOR smtp-out <
TYPE=OUTGOING
ADDRESS=#MX
PROTOCOL=SMTP
INSTANCES=30
SATPOOL_LOOKUP=sat1
>
```

### ***Keyword REDIRECT***

When connectors are one-way only and a pair of connectors are needed in order to both send and receive messages, as is the case for SMPP 3.3 and HTTP connectors, it is now possible to use the REDIRECT keyword. It is used on the receiving connector in order to re-route any outbound messages over to the connector used for sending. For example DLRs are routed back to the connector that requested them if no other routes apply. If that connector is a receive-only connector then the REDIRECT keyword could be used to re-route the outbound messages to a connector that can forward them.

```
CONNECTOR smpp-in <
TYPE=INCOMING
ADDRESS=10.0.0.1:5000
PROTOCOL=SMPP
```

```
INSTANCES=10
USERS=users
# Any outgoing messages routed to this connector should
# be re-routed to smpp-out instead
REDIRECT=smpp-out
>
```

```
CONNECTOR smpp-out <
TYPE=OUTGOING
ADDRESS=10.0.0.2:5000
PROTOCOL=SMPP
INSTANCES=1
>
```

### ***PPG (Push Proxy Gateway) functionality***

The PPG is the component in an MMSC environment that receives WAP Push messages over PAP (Push Access Protocol) and forwards them to a mobile phone over SMS. EMG 3 can handle PAP directly and thus obsoletes the need of a PPG. PAP is an open standard using XML messages over HTTP.

When a PAP message is received it is converted transparently into binary SMS messages which are then forwarded to the SMSC.

If acknowledgements are to be received an outgoing connector is also needed.

Sample connector configuration:

```
CONNECTOR ppg <
TYPE=INCOMING
PROTOCOL=PAP
ROUTE=smsc
>
```

### ***MMS support***

EMG 3 can not only handle e-mail and SMS but also MMS. The following MMS-related protocols are supported both for incoming and outgoing messages: EAIF, MM1, MM7 and CMGPAP. These protocols use SOAP (HTTP+XML) as bearer of data.

Sample MM7 connector configuration:

```
CONNECTOR mm7-out <
TYPE=OUTGOING
ADDRESS=http://mms.example.com:8080/sendmms/
PROTOCOL=MM7
INSTANCES=1
VASPID=myuser
VASID=mysecret
SENDERADDRESS=
>
```

## **Sending MMS from command-line**

It is possible to inject MMS messages using emgsend from command-line. This can be useful for testing. EMG 3 distribution includes the utility mmscomp which compiles a binary MMS message from supplied source files.

Sending a graphics file (test.gif) as MMS:

```
mmscomp -o sample.mms -from +46123456 \  
        -to user@example.com \  
        -subject "Test MMS message" test.gif
```

```
emgsend -port 7186 -file sample.mms +18001234567
```

EMG connectors used for this test:

```
CONNECTOR mgp-mms <  
PROTOCOL=MGP  
TYPE=INCOMING  
ADDRESS=localhost:7186  
INSTANCES=3  
USERS=users  
# Assume MMS content  
DEFAULT_MSGTYPE=MMS  
ROUTE=mms-out  
>
```

```
CONNECTOR mms-out <  
TYPE=OUTGOING  
# Binary encoded MMS  
PROTOCOL=MM1  
ADDRESS=http://localhost:8189/  
INSTANCES=1  
>
```

## **E-mail to MMS**

EMG can perform conversion of e-mail messages to MMS by performing a semantic conversion from the MIME message to an MMS. Any attachments (including an optional SMIL file) will be used. Please note that the attachments must be valid attachments for MMS since EMG does not perform any content validation but only a structural conversion.

## **MMS to e-mail**

Correspondingly incoming MMS can be forwarded as e-mail where the MMS body parts are converted into their MIME equivalents. Once again this is only a semantic conversion. No content conversion is performed.

## ***PDU logs***

While the standard EMG connector logs contain information about messages sent and received as EMG sees them in EMG 3 it is possible to log the actual protocol specific PDUs as well. This is always enabled when using log level DEBUG or DEBUG and can be enabled using the LOGPDU keyword on the connectors for other log levels.

```
CONNECTOR smpp-out <
TYPE=OUTGOING
ADDRESS=10.0.0.1:5000
PROTOCOL=SMPP
INSTANCES=1
LOGPDU
>
```

This will generate a PDU log file name “pdu.smpp-out” in the EMGDIR/log directory.

## ***Message persistence***

Normally messages received and later sent by EMG are only stored in memory (RAM) during their life-time. If the server crashes or terminates abnormally all messages currently in the server could be then lost. Using the separately licensed option message persistence will ensure that all messages are persisted in the file system.

In order to enable message persistence the EMG license must include this option. Also the general keywords SPOOLDIR and PERSISTFILES must be used.

```
SPOOLDIR=/var/spool/emg
PERSISTFILES
```

When running “emgd -v” it will display whether the license has PERSISTFILES enabled:

```
# emgd -v
-----
ENTERPRISE MESSAGING GATEWAY 3.0
-----
Licensed to: Infoflex Connect
Throughput: 10 msgs per second
Serial: emg460000
Hostid: 000d56fec19b
PERSISTFILES available.
EMG SNMP Agent integration enabled.
--
Infoflex Connect AB, Sweden, 2001-2004.
For more info about our messaging products,
please visit http://www.nordicmessaging.se.
```

## ***Plugin API***

It is possible to “hook in” to EMG 3 using an external shared library in order to perform special tasks before and after messages are sent and received by EMG. This makes it possible to implement special billing procedures, advanced routing etc.

The shared library is usually implemented in the C programming language.

Sample connector using the Plugin API:

```
PLUGIN myplugin <
CONFIG=/etc/emg/myplugin.cfg
INSTANCES=1
LIBRARY=/etc/emg/myplugin.so
>
```

```
CONNECTOR smpp-out <
TYPE=OUTGOING
ADDRESS=10.0.0.1:5000
PROTOCOL=SMPP
INSTANCES=1
PLUGIN=myplugin
>
```

The library should export the following functions. All functions are optional, so leave unused functions out instead of having empty stubs.

`void* create_config(char* pluginname, char* configname)`

Load the configuration given in 'configname', and return a pointer to it. This pointer will be used as the first argument to all other functions.

`void destroy_config(void* configuration)`

Free all resources used by the given configuration.

The following four functions are called at different times in the lifetime of a message:

`int before_receive(void* configuration, pluginrequest_t* request,  
                  pluginresponse_t* response)`

Called before a message is accepted into EMG.

If it is ok to proceed it should return 0.

`int before_send(void* configuration, pluginrequest_t* request,  
                  pluginresponse_t* response)`

Called before a message is sent from EMG.

If it is ok to proceed it should return 0.

`int after_send(void* configuration, pluginrequest_t* request,  
                  pluginresponse_t* response)`

Called after a message is sent from EMG.

The status is available in the 'request' struct, in the 'result' field.

```
int after_dlr(void* configuration, pluginrequest_t* request,  
             pluginresponse_t* response)
```

Called after a delivery report is received to EMG.

The status is available in the 'request' struct, in the 'result' field.

For more information about the Plugin API, header files and code samples please contact Infoflex Connect, [info@infoflexconnect.se](mailto:info@infoflexconnect.se).