smsc Documentation

Release stable

Holger Hans Peter Freyther

February 25, 2017

Contents

Author Holger Hans Peter Freyther <holger@moiji-mobile.com>

CHAPTER 1

Introduction

The OsmoSMSC is a scalable Short Message Center implemented using the Pharo Object Environment and MongoDB. Commercial support for deployment, maintenance and extensions is available through moiji-mobile. Please contact help@moiji-mobile.com.

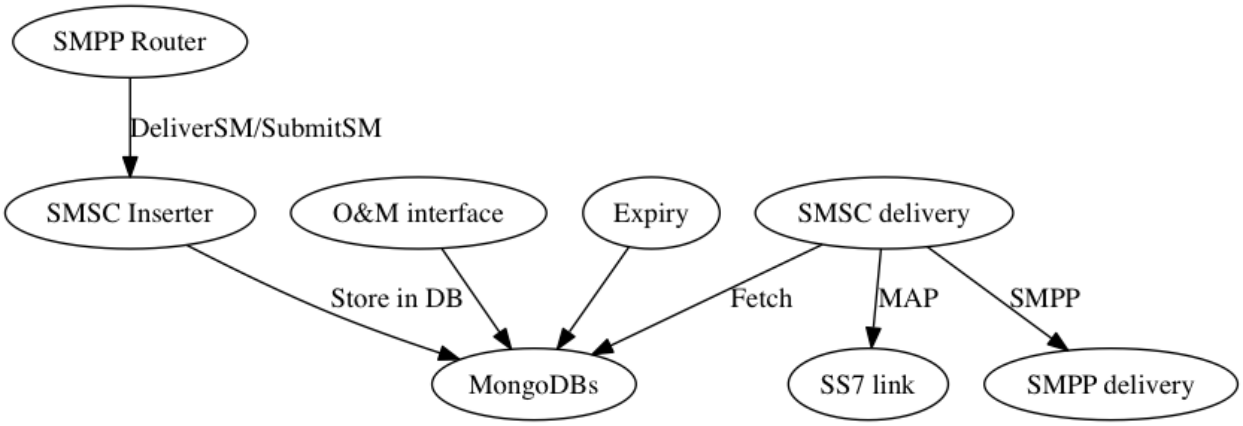
The system consists out of four parts that handle different aspects of the SMS processing. The first step is to configure links and routes on the O&M component through the REST interface, the second is to insert SMPP Deliver_SM, SMPP Submit_SM through SMPP links, the third is the part that handles routing and delivery through SMPP and SS7 and the fourth to expire old SMS and release SMS of failed delivery attempts.

Inserter Provides ESME/MC SMPP links and will store SMPP SubmitSM and SMPP DeliverSM in the database and schedule them for delivery.

Delivery Will determine and route SMS through configured SMPP or SS7 links.

O&M Provides a REST interface to configure SMPP links and SS7 links for the inserter and delivery. Provides an interface to query about the number of queued SMS, failed attempts and more.

GC Expire old queued messages based on the expiration time. Help to clean-up delivery of SMS that got stuck due programming errors on delivery.



CHAPTER 2

Inserter

The inserter is responsible for receiving SMPP DeliverSM and SMPP SubmitSM messages and storing them in the database. Right now no billing is done but it would be done before inserting a message. The message will be re-encoded and stored as such. Conversion will occur on delivery.

Links

One SMPP inserter link can be configured to have a role (ESME or MC) and if it should be a listener or a client connection. The handling of role is not fully implemented at this point in time.

The links will be activated when the inserter application is started, there is no monitoring of changes of the link configuration. For client connections they will be connected and re-connected. If a DNS name is used the hostname will be resolved in a blocking manner. For servers multiple connections for the same systemId are allowed. The SMPP Submit_SM/Deliver_SM response will be sent on the link that initiated the message.

Lifetime

The lifetime of a SMS will be hardcoded to 10 days. The time inside the SMPP message will be ignored. The first delivery will be set to now.

Scaling

Scaling can occur by configuring multiple SMPP links. This way a SMPP message can be decoded while another process is currently waiting for a response of the database. The other approach is to run multiple inserter processes.

Different instances would use the same SMSC Database but a dedicated O&M database to allow the configuration of different links.

CHAPTER 3

Delivery

The task of the delivery is to deliver (or submit) a message. This can be done either by using SMPP or using the GSM Mobile Application Part (MAP). The delivery will attempt to deliver all SMS that are stored in the database.

The deliver functionality will select the SMS that is scheduled next and then will try to find more SMS going to the same destination and will attempt to lock them.

For each SMS a routing decision needs to be made. The routing can be based on the link (the systemId) the SMS arrived, the source address and the destination. Each route can have a list of destinations (e.g. MAPv3, MAPv2, SMPP).

Links

SMPP links can be configured as with the inserter. In addition SS7 links to the osmo-stp can be configured.

Lifetime

The SMS entry in the database contains various fields that control the lifetime. The first is when to expire the SMS and give up. This field will be set by the inserter on insert. The next field is the time the SMS should be delivered. The SMS will be attempted to be submitted/delivered around that time and in the success case will be removed from the database. In the failure case the system will:

- Increase the attempts counter.
- Set a new deliveryTime. Currently this is 30 minutes from now. There is no increasing penalty.

Scaling

Scaling occurs by setting the number of worker threads that determine how many SMS will be tried to submitted/delivered at the same time and by starting multiple instances of the delivery system. The database access is modeled to have only one system to submit/deliver to a given destination at the same time. This will be enforced by a locking scheme in the database and will work with multiple delivery processes.

Routing

The system has two kind of routes. One of them is a default route and the other is a specific route. Each specific route must have a destinationAddressPrefix and can have a sourceAddressPrefix and a systemId. The minimum length for the destinationAddressPrefix is two and if the sourceAddressPrefix is present it must be at least two digits long as well.

To determine a route a database query will be used. The result is sorted by the length of the destinationAddressPrefix, length of the sourceAddressPrefix, the systemId (alphabetically) and if it is a default. The first entry of the result will be picked.

This means the system will not pick a route where most fields match the most but the route that has the longest destinationAddressPrefix and from routes the system with the longest sourceAddressPrefix and if that is the same the system with the matching systemId. The default route will be sorted last.

Route selection examples

In case the following four routes are configured we will look at the route selection of the system with some specific examples.

Route configuration

Default route { “routeName” : “default”, “default”: true }

Matching destination { “routeName” : “destRoute1”, “destinationAddressPrefix” : “49177” }

Matching destination { “routeName” : “destRoute2”, “destinationAddressPrefix” : “491772” }

Matching source and destination { “routeName” : “sourceDestRoute”, “sourceAddressPrefix”: “49166”, “destinationAddressPrefix: “49177” }

Matching systemId, source and destination { “routeName” : “systemIdRoute”, “systemId”: “aSystemId”, “sourceAddressPrefix” : “49166”, “destinationAddressPrefix” : “49177”}

Route selection

Destination 32342435343 There is no match of the destinationAddressPrefix and the route called “default” will be used.

Destination 4917723435 from 49303424324 on link example There are two routes that match these are “destRoute1” and “destRoute2”. The route with the longest matching prefix will be chosen and this is “destRoute2”

Destination 4917723435 from 49166233213 on line example In this case the sourceAddressPrefix of “sourceDestRoute” will match in addition to the destinationAddressPrefix.

Destination 4917723435 from 49166233213 on line aSystemId In this case we have three rules that match the destination and two rules that match the source but only one rule that is matching the systemId. This means the “systemIdRoute” rule will be used.

CHAPTER 4

Expiring messages and Cleaning up

SMS might not succeed to be delivered. Either because the subscriber is not reachable anymore or the delivery might have been interrupted. Either by an administrator restarting the system or a software issue.

Expiring messages

When inserting a SMS the expiration time will be set. The GC application will make a DB query to remove expired SMS atomically. There will be no log statement about which SMS got removed.

Cleaning up

A destination might end-up in a locked state. This can be due a software glitch on delivery or administrator restart. If a destination is locked for more than 30 minutes it will be released.

CHAPTER 5

REST Interface

The inserter, management and the delivery nodes have a dedicated REST interface that is using the GET, PUT and DELETE verbs. The creation of links and routes are idempotent which means creating the same resource will replace the old one. All O&M changes made via the REST interface require a restart of the specific node.

In case more than one node is configured for a specific role one should either use different databases or at least a different collection for the configuration which will require running a different O&M manager as well. Please see the chapter about the CLI arguments on how to do that.

SMPP Inserter Interface

One can configure the SMPPConnection (outgoing connection) and the SMPPConnectionManager (waiting for one connection of a specific systemId/password) through the REST interface. There are operations to list all connections, to create a new one, to look at a specific one and to remove one.

Listing all SMPP connections

```
$ curl -H "Content-Type: application/json" -XGET http://localhost:1700/v1/inserterSMPPLinks
[
        {
                "connectionType" : "client",
                "systemType" : "systemType",
                "password" : "password",
                "role" : null,
                "port" : 4444,
                "hostname" : "nameOfServer",
                "systemId" : "systemId",
                "connectionName" : "NAME"
        },
        {
                "connectionType" : "server",
                "systemType" : "systemType",
                "allowedRemotePort" : 6666,
                "allowedRemoteAddress" : "127.0.0.1",
                "password" : "password",
                "port" : 5555,
                "role" : null,
                "systemId" : "systemId",
                "connectionName" : "NAME2"
```

```
        }
]
```

Result codes Under normal operation only 200 with an JSON array should be returned.

Creating or updating a SMPP connection

The SMPPConnection of type "client" can specify the remote hostname and port while the SMPPConnectionManager of type "server" allows to specify the port to bind to and from which remote IPv4/port the connection should arrive.

```
$ curl -H "Content-Type: application/json" -XPUT http://localhost:1700/v1/inserterSMPPLink/NAME \
-d '{
        "connectionType": "client",
        "hostname": "nameOfServer",
        "port": PortNumber,
        "systemId": "systemId",
        "systemType": "systemType",
        "password": "password"
}'
"OK"

$ curl -H "Content-Type: application/json" -XPUT http://localhost:1700/v1/inserterSMPPLink/NAME2 \
-d '{
        "connectionType": "server",
        "port": PortNumber,
        "systemId": "systemId",
        "systemType": "systemType",
        "password": "password",
        "allowedRemoteAddress": "127.0.0.1",
        "allowedRemotePort": aSourcePortNumber
}'
"OK"
```

Result codes In case of invalid JSON a 5XX response will be returned, in case of incomplete document a 5XX will be returned as well, in case no connection can be created a 4XX will be returned

Inspect a SMPP connection

Show the settings of one configured SMPP link. This can either be a client or server.

```
$ curl -H "Content-Type: application/json" -XGET http://localhost:1700/v1/inserterSMPPLink/NAME
{
        "connectionType" : "client",
        "systemType" : "systemType",
        "password" : "password",
        "role" : null,
        "port" : 4444,
        "hostname" : "nameOfServer",
        "systemId" : "systemId",
        "connectionName" : "NAME"
}
```

Result codes In case no connection with than name exists a 404 will be returned, otherwise a 200 with the JSON response response will be returned

Delete a SMPP connection

Remove the configuration of a SMPP link.

```
$ curl -H "Content-Type: application/json" -XDELETE http://localhost:1700/v1/inserterSMPPLink/NAME
"OK"
```

Result codes In case no connection with name exists a 404 will be returned, otherwise a 200 with an EMPTY return will be returned.

SMPP Delivery Interface

It is possible to make deliveries using SMPP. These links are configured independly to the inserter interface but follow the same documents as with the inserter, the only difference is the URL.

Instead of inserterSMPPLink it is deliverySMPPLink and instead of inserterSMPPLinks it is deliverySMPPLinks.

— Parameters same as with the inserter $ curl -H "Content-Type: application/json" -XGET http://localhost:1700/v1/deliverySMPPLinks $ curl -H "Content-Type: application/json" -XPUT http://localhost:1700/v1/deliverSMPPLink/NAME $ curl -H "Content-Type: application/json" -XGET http://localhost:1700/v1/deliverSMPPLink/NAME $ curl -H "Content-Type: application/json" -XDELETE http://localhost:1700/v1/deliverSMPPLink/NAME ...

SS7 Delivery Interface

The main function of the SMSC Delivery is to deliver using SS7. One needs to configure one or multiple network connections to the osmo-stp SCTP/TCP bridge. The configuration is very similar to the above routines and supports the same verbs.

Listing all SS7 Network Services

```
$ curl -H "Content-Type: application/json" -XGET http://localhost:1700/v1/deliverySS7Links
[
        {
                "class" : "SCCPNetworkServiceOsmoDirect",
                "token" : "Token",
                "port" : 12345,
                "connectionName" : "NAME",
                "hostname" : "host"
        }
]
```

Creating a SS7 Network Service

```
$ curl -H "Content-Type: application/json" -XPUT http://localhost:1700/v1/deliverySS7Link/NAME \
-d '{
        "hostname": "host",
        "port": PortNumber,
        "token": "Token"
}'
"OK"
```

Inpect a SS7 Network Service

Show the settings of one configured SS7 delivery link.

— $ curl -H “Content-Type: application/json” -XGET http://localhost:1700/v1/deliverySS7Link/NAME { “class” : “SCCPNetworkServiceOsmoDirect”, “token” : “Token”, “port” : 12345, “connectionName” : “NAME”, “hostname” : “host” } —

Delete a SS7 Network Service

Delete a configured SS7 delivery link.

```
$ curl -H "Content-Type: application/json" -XDELETE http://localhost:1700/v1/deliverySS7Link/NAME
"OK"
```

Routes for Delivery

A route is looked-up before the delivery of a SMS is attempted. The next sections list commands to query and manipulate routes.

Listing all routes

```
$ curl -H "Content-Type: application/json" -XGET http://localhost:1700/v1/deliveryRoutes

[
    {
        "systemId" : "OptionalSystemdIdMatch",
        "default" : false,
        "destinationAddressPrefixLength" : 4,
        "priority" : 100,
        "destinationAddressPrefix" : "1234",
        "sourceAddressPrefix" : "4567",
        "methods" : [
            {
                "connectionName" : "NAME",
                "class" : "ShortMessageDeliveryMethodSMPP",
                "messageType" : "deliverSM"
            },
            {
                "class" : "ShortMessageDeliveryMethodSS7",
                "ssn" : 7,
                "globalTitle" : "49111111",
                "sendRoutingInfoTranslationType" : 2,
                "smscNumber" : "49111111",
                "forwardSMTranslationType" : 0,
                "connectionName" : "NAME",
                "mapVersion" : 2
            }
        ],
        "routeName" : "NAME",
        "sourceAddressPrefixLength" : 4
    }
]
```

Result codes Under normal operation only 200 with an JSON array should be returned.

Creating or updating a route

```
$ curl -H "Content-Type: application/json" -XPUT http://localhost:1700/v1/deliveryRoute/NAME \
-d '{
    "systemId": "OptionalSystemdIdMatch",
    "priority": OptionalNumberPriority,
    "default": OptionalBooleanDefault,
    "destinationAddressPrefix": "OptionalDestinationPrefix",
    "sourceAddressPrefix": "OptionalSourceAddressPrefix",
    "methods": [
        {
            "class": "ShortMessageDeliveryMethodSMPP",
            "connectionName": "aSMPPDeliveryLinkName",
            "messageType": "SMPPMessageTypeToUse"
        },
        {
            "class": "ShortMessageDeliveryMethodSS7",
            "connectionName": "aSS7DeliveryLinkName",
            "globalTitle": "CallingGT to use",
            "ssn": aCallingSsnNumber,
            "smscNumber": "aSMSCGTNumber",
            "sendRoutingInfoTranslationType": aGTTranslationType,
            "forwardSMTranslationType": aGTTtranslationType,
            "mapVersion": aVersionNumber
        }
    ]
}'
"OK"
```

messageType Either deliverSM or submitSM are valid for class ShortMessageDeliveryMethodSMPP.

Result codes In case of invalid JSON a 5XX response will be returned, in case of incomplete document a 5XX will be returned as well, in case no connection can be created a 4XX will be returned

Inspect a route

Show the settings of one configured SMPP link. This can either be a client or server.

```
$ curl -H "Content-Type: application/json" -XGET http://localhost:1700/v1/deliveryRoute/NAME
{
    "systemId" : "OptionalSystemdIdMatch",
    "default" : false,
    "destinationAddressPrefixLength" : 4,
    "priority" : 100,
    "destinationAddressPrefix" : "1234",
    "sourceAddressPrefix" : "4567",
    "methods" : [
        {
            "connectionName" : "NAME",
            "class" : "ShortMessageDeliveryMethodSMPP",
            "messageType" : "deliverSM"
        },
        {
            "class" : "ShortMessageDeliveryMethodSS7",
```

```
            "ssn" : 7,
            "globalTitle" : "49111111",
            "sendRoutingInfoTranslationType" : 2,
            "smscNumber" : "49111111",
            "forwardSMTranslationType" : 0,
            "connectionName" : "NAME",
            "mapVersion" : 2
        }
    ],
    "routeName" : "NAME",
    "sourceAddressPrefixLength" : 4
}
```

Result codes In case no connection with than name exists a 404 will be returned, otherwise a 200 with the JSON response response will be returned

Delete a route

```
$ curl -H "Content-Type: application/json" -XDELETE http://localhost:1700/v1/deliveryRoute/NAME
"OK"
```

Result codes In case no connection with name exists a 404 will be returned, otherwise a 200 with an EMPTY return will be returned.

CHAPTER 6

Command Line Interface

The system installs templates that combined with the Pharo image-launch allows to configure and start the images in the right configuration.

Common options

--db-host The hostname of the mongo database system

--db-port The port of the mongo database system

--statsd-host The hostname/IPv4 address to use for statsd.

--statsd-port Use if –statsd-host has been supplied and determines the target address for the statsD server.

--smscdb-name The name of the SMSC database to use

--omdb-name The name of the O&M database to use

O&M image

--rest-port The port to use to expose the REST interface

Inserter image

No specific options.

Delivery image

--jobs The number of jobs that process and send SMS. This controls the concurrency of the delivery.

GC image

TODO