SMS Module

Bogdan Iancu
FhG FOKUS

Edited by

Bogdan Iancu

SMS Module

Edited by and Bogdan Iancuand Bogdan Iancu

Revision History

Revision $Revision: 1.1 $ $Date: 2003/07/23 16:19:31 $

Table of Contents

List of Examples

Chapter 1. User’s Guide

1.1. Overview

This module provides a way of comunication between SIP network (via SIP MESSAGE) and GSM networks (via ShortMessageService). Comunication is possible from SIP to SMS and vice versa. The module provides facilities like SMS confirmation--the gateway can cofirm to the SIP user if his message really reached its destination as a SMS--or multipart messages--if a SIP messages is too long it will be split and sent as multiple SMS.

Errors occurred because of an invalid number or a too long message or because of an internal modem malfunction are reported back to the SIP user via a SIP message containing explanations regarding the error.

1.1.1. Hardware Requirements

The SMS module needs a GSM modem to be able to send/receive the SMS messages. Usualy, this kind of modems are externals, linked to the machine via serial cable. The modem can be a dedicated one (as the ones provided by FALCOM) or can be a GSM telephone that has an internal modem (as the latest mobile phones from NOKIA and ERICSSON).

1.1.2. Numbering Plan

The gateway accepts and advertises phone numbers in international format, more specific like: +(international code)(area code)(number). Ex: Germany, D1 = +49 170 5678181 Romania, Connex = +40 722 123456 A number in this format is expected to be placed as username into RURI or in the To header. If RURI misses the username, the To header will be consider. Also, the gateway will advertise in this format the username in Contact headers (in SIP replies and requests) and in From headers (in SIP requests).

1.1.3. Address Mapping

To identify the destination number of the SMS, the gateway expects to have a mobile number in username of the SIP destination address (for example sip:+401704678811@iptel.org). For the reverse direction, because the gateway has only one GSM number, the destination SIP address has to be encapsulated into the SMS body. The gateway expects to find a SIP address at the beginning of the SMS body in “sip:user.host” format. Everything before the SIP address will be discarded, the useful text begins exactly after the address (for example SMS=“For sip:user@host hello world!!” -> SIP=“hello world”) In order to facilitate replying, the gateway sends all the SMS messages with a header containing the source SIP address in the following format: “From sip:user@host (if you reply DONOT remove it)<new_line>”. When an SMS-reply is received having this header (all of it or truncated at the end), the header will be left out (it will not be in the SIP message).

1.2. Dependencies

1.2.1. SER Modules

The following modules must be loaded before this module:

- *tm - Transaction Manager*.

1.2.2. External Libraries or Applications

The following libraries or applications must be installed before running SER with this module loaded:

- *None*.

1.3. Exported Parameters

1.3.1. `modems` (string)

Define and configure one or more GSM modems.

```
modems_value     = modem_definition *( ";" modem_definition )
modem_definition = modem_name "[" list_of_params "]"
list_of_params   = modem_param *( ";" modem_param )
modem_param      = name "=" value
```

The following parameters can be used:

- d=device (mandatory) - Device associatd with modem (/dev/ttyS0, /dev/modem, etc.).
- p=pin (optional) - SIM PIN - default is NULL.
- m=mode (optional) - Modem working mode ("ASCII","OLD","DIGICOM","NEW"). Default value is "NEW".
- c=SMS_Center (optional) - SMS center number for that modem. Deafult is the SMS center set on the SIM card.
- b=baudrate (optional) - Default is 19600.
- r=retry (optional) - How many times to try to re-send a SMS that reported error. Default is twice.
- l=looping (optional) - Time for modem to wait before performing a new check for incomimg/outgoing SMS/SIP_MSG. Default is 20.

No default value, the parameter is mandatory.

Example 1-1. Set `modems` parameter

```
...
modparam("sms", "modems", "Nokia [d=/dev/ttyS1;b=9600;m=new;l=30] ")
modparam("sms", "modems", "Nokia[d=/dev/ttyS1];Siemens[d=/dev/ttyS2]")
...
```

1.3.2. `networks` (string)

Define and configure used GSM networks.

```
networks_value = net_definition *( ";" net_definition )
net_definition = net_name "[" list_of_params "]"
list_of_params = set_param *( ";" set_param )
set_param      = name "=" value
```

The following parameters can be used:

- m=msx_sms_per_call (optional) - Maximum number of SMS send / received from that net in one modem loop. Default is 10. This parameter was introduced to avoid starvation.

  Example of the starvation--a modem can send SMS for more than 1 networks. If you have a huge number of SMS for the first network and the number of incoming SIP messages is equal to the sent SMS per same unit of time, the modem will never get to send SMS for the next networks.

No default value, the parameter is mandatory.

Example 1-2. Set `networks` parameter

```
...
modparam("sms", "networks", "D1 [m=10] ;d2[ m=20]")
...
```

1.3.3. `links` (string)

Define from which network each modem should send SMS.

```
links_value = modem_assoc *( ";" modem_assoc )
modem_assoc = modem_name "[" list_of_networks "]"
list_of_networks = network *( ";" network )
```

No default value, the parameter is mandatory.

Example 1-3. Set `links` parameter

```
...
modparam("sms", "links", "NOKIA[D1;d2]")
...
```

The modem NOKIA will send SMS from D1 and D2 net (in this order !). if in a net queue are more then max_sms_per_call SMS the modem will *not sleep* before starting the next loop ! Shortly, if messages are waiting to be sent, the modem will not go in sleep.

1.3.4. `default_net` (string)

The default network to use. If no one specified, the first defined network is used. This parameter is useful only if the the "sms_send_msg" exported function is used (see Section 1.4).

Example 1-4. Set `default_net` parameter

```
...
modparam("sms", "default_net", "D1")
...
```

1.3.5. `max_sms_parts` (integer)

Shows in how many parts (SMS messages) a SIP message can be split. If exceeded, the SIP message will be sent truncated and the SIP user will get back another message containing the unsent part.

Default value is 4.

Example 1-5. Set `max_sms_parts` parameter

```
...
modparam("sms", "max_sms_parts", 10)
...
```

1.3.6. `domain_str` (string)

Specify a fake domain name to be used by the gateway. The Contact headers and the From header from request will be construct based on this fake domain name. It's useful when the gateway is transparently hidden behind a proxy/register (located on different machines).

Default is the name of the machine the gateway is running on.

Example 1-6. Set `domain_str` parameter

```
...
modparam("sms", "domain_str", "foo.bar")
...
```

1.3.7. `use_contact` (integer)

If a contact header should be added to the outgoing SIP messages. Even if the SIP draft forbids this, some UAS require it.

Default is 0 (no).

Example 1-7. Set `use_contact` parameter

```
...
modparam("sms", "use_contact", 1)
...
```

1.3.8. `sms_report_type` (integer)

If the modem should ask for SMS confirmation from the SMS Center. If the SMSC reply with an error code, the gateway will send back to SIP user a SIP message containing the text (or part of it) that couldn't be send. Two report mechanisms are implemented:

- 1 - the reports are delivered by the GSM device as SMS reports (so far supported only by Nokia modems);
- 2 - the reports are delivered as async. CDS reponses (supported by almost all modems, exepting Ericsson).

Default is 0 (no report).

Example 1-8. Set `sms_report_type` parameter

```
...
modparam("sms", "sms_report_type", 1)
...
```

1.4. Exported Functions

1.4.1. `sms_send_msg_to_net(network_name)`

Put the SIP msg in the specified network queue. The function return error if the number encapsulated into SIP message is malformed, if the content_type is incorrect or because of some internal failures.

Meaning of the parameters is as follows:

- *network_name* - Name of network.

Example 1-9. `sms_send_msg_to_net` usage

```
...
if (sms_send_msg_to_net("D1"))
{
    if (!t_reply("202", "yes sir, SMS sent over"))
    {
        # if replying failed, retry statelessly
        sl_reply_error();
    };
} else {
    if (!t_reply("502", "Bad gateway - SMS error"))
    {
        # if replying failed, retry statelessly
        sl_reply_error();
    };
    break;
};
...
```

1.4.2. `sms_send_msg()`

The same as the previous one, but use the default network queue.

Example 1-10. `sms_send_msg` usage

```
...
if (sms_send_msg_to_net())
{
    if (!t_reply("202", "yes sir, SMS sent over"))
    {
        # if replying failed, retry statelessly
        sl_reply_error();
    };
} else {
    if (!t_reply("502", "Bad gateway - SMS error"))
    {
        # if replying failed, retry statelessly
        sl_reply_error();
    };
    break;
};
...
```

Chapter 2. Developer’s Guide

Each modem forks its own process for sending /fetching SMS. Communication and queuing between ser working processes and modem processes is done with pipes.

Chapter 3. Frequently Asked Questions

1. Where can I find more about SER?

Take a look at http://iptel.org/ser.

2. Where can I post a question about this module?

First at all check if your question was already answered on one of our mailing lists:

- http://mail.iptel.org/mailman/listinfo/serusers
- http://mail.iptel.org/mailman/listinfo/serdev

E-mails regarding any stable version should be sent to <`serusers@iptel.org`> and e-mail regarding development versions or CVS snapshots should be send to <`serdev@iptel.org`>.

If you want to keep the mail private, send it to <`serhelp@iptel.org`>.

3. How can I report a bug?

Please follow the guidelines provided at: http://iptel.org/ser/bugs