mod_gsmopen

About

GSMopen is a FreeSWITCH™ endpoint (channel driver) that allows an SMS message to be sent to and from FreeSWITCH as well as incoming and outgoing GSM voice calls that can be bridged, originated, answered, etc. as in all other endpoints, e.g. sofia/SIP). An SMS on FreeSWITCH is handled following the CHAT API (like the text messaging in mod_sofia, mod_skypopen, and mod_dingaling).

Status
GSMopen is in “beta” status.

Quick Start Guide

Steps needed to use GSMopen:

- Compile and install FreeSWITCH
- Compile and install the prerequisites (gsmlib and libctb)
- Compile and install mod_gsmopen
- Install and edit mod_gsmopen config file
- Connect one or more Huawei USB dongles (or other GSM modems) to the FreeSWITCH server machine
- Be sure the dongle has the “voice” capability UNLOCKED, or unlock it with dc-unlocker (http://www.dc-unlocker.com/).
- Start FreeSWITCH
- Load mod_gsmopen in FreeSWITCH
- Profit!

What Is GSMopen

GSMopen is an endpoint (channel driver) that allows an SMS to be sent to and from FreeSWITCH as well as incoming and outgoing GSM voice calls (that can be bridged, originated, answered, etc. as in all other endpoints, e.g. sofia/SIP). An SMS on FreeSWITCH is handled following the CHAT API (like the text messaging in mod_sofia, mod_skypopen, and mod_dingaling).

Preferred GSM modem to use for voice calls (and SMS) is Huawei E1550 dongle, or compatible.

GSMopen works in FreeSWITCH on Linux and Windows, native at 8kHz (GSM is 8kHz compressed audio). It probably works on *BSD and OSX too.

GSMopen operates on GSM cellular/mobile connections in the same way that OpenZAP operates on analog lines. One interface (a GSM modem) is needed for each channel. For example, two concurrent calls would need two channels as well as two USB GSM dongles connected to the FreeSWITCH server box.

Obviously you must have credit on the SIM(s) inside the modems to make and receive calls, just like you need credits for your regular cell phone to work.

GSMopen uses very low CPU, so it works with the less powerful server platforms without problems (e.g. embedded appliances).

GSMopen has been contributed to the community by: Giovanni Maruzzelli (gmaruzz at gmail dot com) with a lot of help from the core developer team and hints, patches, suggestions, bug reports, feature requests from the superlative FS community.

GSMopen is fully integrated with Mod_sms so you can use it for your messaging system pleasure.

DATA CONNECTION Concurrent on Same Dongle
Tarmo Aia wrote: The only working (voice) dongle what i have tested is E173, with that there is no need to to anything extraordinary. Just standard setup. The tests reveal that if You have 3G coverage, the data and voice would be concurrent, if you have below 3G (2,5 or 2 or 1G) in the very moment when you start voice calling over GSM dongle, the data would be disconnected. The data connection would be restored after the call is hung up - actually this is very logical because when the dongle is using 2G or similar coverage, the frequency for calling and data transmitting is the same. I suspect that if you call with 3G coverage, the call would go in the 1G frequency. If you are more interested I can put the dongle to the radio lab and do some measurements ;-) 

About setup: I'm using FS 1.2-stable, for bringing up data connection i use wvdial, config would be like: (/etc/wvdial.conf)

```
[Dialer xxx]
Modem = /dev/ttyUSB0
Baud = 115200
Init2 = AT+CGDCONT=1,"IP","internet.xxx.ee"
Phone = *99#
Username = *
Password = *
New PPPD = yes
Auto DNS = 1
```

To run data just type wvdial xxx (the xxx should be your provider or smth)

In the test machine, config files for FS were pretty out-of-the-box, in gsmopen.conf i just changed the interface control ports to right ones. Hope it was enough information,

**Hardware Requirements**

The CPU load generated by the GSMopen endpoint is very low, so if a server is able to run FS, it will have no problems using GSMopen channels. You will need at least one physical interface to connect to a GSM network.

**Compatibility List**

Preferred GSM modem to use with GSMopen is Huawei E1550 dongle, or compatible. Please add to this list; probably most dongles are compatible.

- E1550
- E1552
- E169
- E1692
- E171
- E173
- E175X
- E1762C
- E1762
- E180
- K3520
- K3715
- K3765

Look for "Huawei modem" on Ebay or similar sites. Eg: http://www.aliexpress.com/wholesale?isFreeShip=y&SearchText=huawei%2Be1550&CatId=0&manual=y

Anyway, if other brands or models have the basic USB interfaces (serial + audio) but requires different modem commands, I will add support to them. Open a Jira for it, or contact me directly.

**USB Power**

MORE THAN ONE DEVICE on an USB bus without dedicated power supply can have intermittent failure, because dongles can use all of the power! If you use more than one device, use an external power supply connected to a.c. mains to supply your USB hub. Each dongle requires about 500mA / 0.5W to be safe.

**Voice calls and SMS**
If you have trouble with voice calls, please check that the dongle has the "voice" capability and that "voice" capability is unlocked. To check the capability existence and status, and unlock if needed, you can use dc-unlocker client for Windows (http://www.dc-unlocker.com), client and checking are free, small fee to unlock.

SMS ONLY (no voice calls)

- Any GSM modem (secondhand cellphone or a professional modem) that accepts AT commands and its data cable to connect it to the server.

Dialplan

How to use GSMopen for outbound voice calls from FreeSWITCH™.

Like other endpoints it's easy to build up useful dialplans using GSMopen. You can use the standard format with the interface name:
gsmopen/interface1/3472665618

To call the number "3472665618" using the gsmopen interface named "interface1"

Dialplan snippet:

<table>
<thead>
<tr>
<th>GSMopen Dialplan Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!-- dial 3472665618 via gsmopen using interface1 interface to go out --&gt;</td>
</tr>
<tr>
<td>&lt;extension name=&quot;gsmopen&quot;&gt;</td>
</tr>
<tr>
<td>&lt;condition field=&quot;destination_number&quot; expression=&quot;^2909$&quot;&gt;</td>
</tr>
<tr>
<td>&lt;action application=&quot;bridge&quot; data=&quot;gsmopen/interface1/3472665618&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/condition&gt;</td>
</tr>
<tr>
<td>&lt;/extension&gt;</td>
</tr>
</tbody>
</table>

The "ANY" and "RR" interfaces

The poor man's interface grouping.

You can also use the "ANY" or "RR" interfaces
gsmopen/ANY/3472665618
gsmopen/RR/3472665618

Use the ANY alias to call "3472665618" using the first available (idle, not in a call) gsmopen interface, automatically selected (thx Seven Du).

Use the RR alias to select the interface using a Round Robin algorithm to distribute calls more fairly between all the available interfaces.

Dialplan snippet:

<table>
<thead>
<tr>
<th>GSMopen Round Robin Dialplan Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!-- dial 3472665618 via gsmopen RR interface --&gt;</td>
</tr>
<tr>
<td>&lt;extension name=&quot;gsmopen&quot;&gt;</td>
</tr>
<tr>
<td>&lt;condition field=&quot;destination_number&quot; expression=&quot;^2908$&quot;&gt;</td>
</tr>
<tr>
<td>&lt;action application=&quot;bridge&quot; data=&quot;gsmopen/RR/3472665618&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;/condition&gt;</td>
</tr>
<tr>
<td>&lt;/extension&gt;</td>
</tr>
</tbody>
</table>

CONFIGURATION FILE and incoming voice calls

IMEI and IMSI automatic device discovery (only on Linux)
These two new configuration parameters allow for startup automatic discovery and configuration of command (controldevice_name) and audio (controldevice_audio_name) devices.

Eg: in an interface definition you can specify the IMEI of the dongle, or the IMSI of the SIM card contained into the dongle, or both, and mod_gsmopen will scan the system looking for the correct devices.

Values automatically discovered at startup override controldevice_name and controldevice_audio_name. This means that if discovery works you can omit controldevice_name and controldevice_audio_name from the interface definition.

If BOTH IMEI and IMSI are specified in one interface description, they MUST be BOTH correct. You can specify just one, if you prefer.

**Configuration File**

GSMopen is very configurable.

Almost any single AT command used to manage call flow and to understand signaling and status can be customized.

There are default values for all values, so you can leave the configuration file almost empty (you lazy!).

```
<configuration name="gsmopen.conf" description="GSMopen Configuration">
  <global_settings>
    <param name="debug" value="8"/>
    <param name="dialplan" value="XML"/>
    <param name="context" value="default"/>
    <param name="hold-music" value="$$\{moh_uri\}"/>
    <param name="destination" value="9999"/>
  </global_settings>
  <!-- one entry here per gsmopen interface -->
  <per_interface_settings>
    <interface id="1" name="interface0">
      <param name="hold-music" value="$$\{moh_uri\}"/>
      <param name="dialplan" value="XML"/>
      <param name="context" value="default"/>
      <param name="destination" value="5000"/>
      <param name="controldevice_name" value="/dev/ttyUSB3"/>
      <param name="controldevice_audio_name" value="/dev/ttyUSB2"/>
      <param name="imei" value="353443043468086"/>
      <param name="imsi" value="220032500058601"/>
    </interface>
    <interface id="3" name="interfaceNICE">
      <param name="hold-music" value="$$\{moh_uri\}"/>
      <param name="dialplan" value="XML"/>
      <param name="context" value="default"/>
      <param name="destination" value="9996"/>
      <param name="controldevice_name" value="/dev/ttyUSB7"/>
      <param name="controldevice_audio_name" value="/dev/ttyUSB6"/>
    </interface>
  </per_interface_settings>
</configuration>
```

Following are all the various configurable parameters you can set for each interface (with their default values):

**GSMopen Defaults**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>dialplan</td>
<td>&quot;XML&quot;</td>
</tr>
<tr>
<td>destination</td>
<td>&quot;5000&quot;</td>
</tr>
<tr>
<td>controldevice_name</td>
<td>NULL</td>
</tr>
<tr>
<td>controldevice_audio_name</td>
<td>NULL</td>
</tr>
<tr>
<td>digit_timeout</td>
<td>NULL</td>
</tr>
<tr>
<td>max_digits</td>
<td>NULL</td>
</tr>
<tr>
<td>hotline</td>
<td>NULL</td>
</tr>
<tr>
<td>dial_regex</td>
<td>NULL</td>
</tr>
</tbody>
</table>
hold_music = NULL
fail_dial_regex = NULL
enable_callerid = "true"
at_dial_pre_number = "ATD"
at_dial_post_number = ";"
at_dial_expect = "OK"
at_answer = "ATH"
at_answer_expect = "OK"
at_send_dtmf = "AT+VTS"
at_preinit_1 = ""
at_preinit_1_expect = ""
at_preinit_2 = ""
at_preinit_2_expect = ""
at_preinit_3 = ""
at_preinit_3_expect = ""
at_preinit_4 = ""
at_preinit_4_expect = ""
at_preinit_5 = ""
at_preinit_5_expect = ""
at_postinit_1 = "at+cmic=0,0"
at_postinit_1_expect = "OK"
at_postinit_2 = "AT+CKPD="EEE"
at_postinit_2_expect = "OK"
at_postinit_3 = "AT+CSSN=1,0"
at_postinit_3_expect = "OK"
at_postinit_4 = "at+sidet=0"
at_postinit_4_expect = "OK"
at_postinit_5 = "at+clvl=99"
at_postinit_5_expect = "OK"
at_query_battchg = "AT+CBC"
at_query_battchg_expect = "OK"
at_query_signal = "AT+CSQ"
at_query_signal_expect = "OK"
at_idle = "+MCST: 1"
at_incoming = "+MCST: 2"
at_active = "+CSSI: 7"
at_failed = "+MCST: 65"
at_calling = "+CSSI: 1"
at_indicator_noservice_string = "CIEV: 20"
at_indicator_nosignal_string = "CIEV: 50"
at_indicator_lowsignal_string = "CIEV: 5;1"
at_indicator_lowbattchg_string = "CIEV: 0;1"
at_indicator_nobattchg_string = "CIEV: 0;0"
at_indicator_callactive_string = "CIEV: 3;1"
at_indicator_nocallsetup_string = "CIEV: 3;0"
at_indicator_callsetupincoming_string = "CIEV: 6;1"
at_indicator_callsetupoutgoing_string = "CIEV: 6;2"
at_indicator_callsetupremotering_string = "CIEV: 6;3"
alsacname = "plughw:1"
alsapname = "plughw:1"
at_early_audio = "0"
at_after_preinit_pause = "500000"
at_initial_pause = "500000"
at_has_clcc = "0"
at_has_ecam = "0"
alsa_period_size = "160"
alsa_periods_in_buffer = "4"
gsmopen_sound_rate = "8000"
gsmopen_serial_sync_period = "300"
Incoming Voice Calls

Each incoming voice call that arrives on the interface will be directed to the destination extension in the context context.

So, please edit or add those fields to the config file to adapt it to your needs. The default config file works with the default, out-of-the-box, demo FreeSWITCH dialplan.

Multiple Concurrent Incoming Calls to the Same GSM Number

This is possible, if the carrier supports call forwarding. You must set up call forwarding on BUSY, NOT REACHABLE state. Remember to switch off CALL WAITING.

Each physical interface (eg: GSM modem) has its own SIM, with just one number. You must have one interface for each concurrent call.

When a call is made by a remote party to a number, the carrier sends the call to the SIM that bears that number. If that SIM is busy or unreachable, the carrier will redirect the call to the forwarded number. Same way you can add additional SIM cards / phones - forward them.

The carrier can limit maximum call forwarding chain length. If you experience that situation, please add that information here.

API and CLI Commands

GSMopen adds various commands to the standard FreeSWITCH API and Commands.

They can all be used both through the command line and via API / ESL / whatever.

Integration with mod_sms

GSMopen is fully integrated with mod_sms so you can use it for your messaging system pleasure.

GSM Commands

"gsm" commands are intended to be used from the FS command line ("gsm remove" and "gsm reload" can be useful from Event socket as well).

console interface_name

You begin typing "gsm console interface_name" to direct the "current console" to sending messages to interface_name. From now on, you can type "gsm AT_command" and AT_command string will be sent to the modem identified by interface interface_name.

list

"gsm list" gives the list and status of all the running GSMopen interfaces (a star marks the interface from which "RR" - see below - will start hunting an IDLE one), statistics about inbound failed and total calls, outbound failed and total calls per each interface.

remove

This command removes the gsmopen interface with name interface_name or with id interface_id, if that interface is idle.

reload

This command re-reads GSMopen configuration file gsmopen.conf.xml and adds ONLY the non running interfaces it found in that file. All existing running interfaces are not affected.
gsmopen Commands

"gsm remove" and "gsm reload" (see above) can be useful from API / ESL / whatever as well.

"gsmopen" commands are intended to be used by programs (API / ESL / whatever) and have the format: "gsmopen interface_name AT_command_string". They send the AT_command string to the modem identified by interface_name.

gsmopen interface2 ATI

This allows you to use directly the entire power of the AT command set of your GSM modem or cellphone. For example, to prototype a new feature, do customization, etc. Typing "console loglevel 9" at the FS command line allows you to see the AT answers from the GSM modem.

**gsmopen_boost_audio**

This command affects the volumes of incoming and outbound audio at the sample level, in code. This command may be useful to interactively (trial and error during a call) find the best audio gain setting for your setup, then you write the found values in the config file. Boost can be for playback or for capture, and can be negative or positive (expressed in deciBel units). Syntax is:

```
gsmopen_boost_audio interface_name [play|capt] <value_in_deciBels>
```

where the boost value can range from -40dB to +40dB

Example:

gsmopen_boost_audio interface3 play -10

The example will lower by 10 decibels the volume of the playback in interface3

**gsmopen_dump**

This command generates (fires) a CUSTOM event of type gsmopen::dump_event that reports a lot of useful information about the interface interface_name.

If interface_name is "list", gsmopen_dump will fire as many events as the number of running interfaces, one for each of them.

gsmopen_dump interface1

Or,

gsmopen_dump list

For the event description, see below, **Events**.

**gsmopen_sendsms**

It supports full UTF8 SMS text, although the FS command line only accepts ASCII. Please use ESL or API to send UTF8 text.

gsmopen_sendsms interface_name destination_number SMS_text

Example:

gsmopen_sendsms interface1 3472665618 this is a nice SMS text

**chat**

GSMopen answers to the FreeSWITCH standard "chat" command too, and uses its arguments to execute a gsmopen_sendsms command. So, if you have a messaging application that uses the chat command with Sofia/SIP or Jingle, no need to recode it with special cases for SMS messages :-)

It uses SMS as protocol specification. e.g., from command line:

```
chat SMS|interface3|3472665618|ciao amore
```

**ussd on chat**

You can send ussd to operator, using the special destination "ussd". E.g:

```
chat SMS|interface3|ussd|*141#
```

**Events**

GSMopen generates (fires) various CUSTOM events in addition to the standard FreeSWITCH events on voice calls (like the other endpoints) and MESSAGE (chat) events on incoming SMS (like Sofia and Jingle).

**Voice Calls**
Standard CODEC and CHANNEL_* events.

See Event list

**MESSAGE (SMS)**

The Event type generated by an incoming SMS is of type MESSAGE (like in Jingle and Sofia).

The interesting fields are:

<table>
<thead>
<tr>
<th>GSMopen Event Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>login: the interface name that received the SMS</td>
</tr>
<tr>
<td>from: the sender’s number, urlencoded</td>
</tr>
<tr>
<td>date: the date of the received message, urlencoded</td>
</tr>
<tr>
<td>datacodingscheme: which kind of alphabet was used to send the message</td>
</tr>
<tr>
<td>servicecentreaddress: address of SC used to send the message</td>
</tr>
<tr>
<td>messagetype: numeric, usually 0, kind of message</td>
</tr>
<tr>
<td>during-call: bool, the message was received while a voice call was ongoing?</td>
</tr>
</tbody>
</table>

And obviously the body, encoded in UTF8, that contains the SMS text.

**UTF8 Encoding**

The body is UTF8 encoded, gives you ASCII for ASCII, and UTF8 for all the rest.

This is a telnet session to the Events port (8021) asking for authorization, asking for events of type message in plain format, then an incoming SMS as reported with Events plain.
GSMopen Event Query

telnet 127.0.0.1 8021
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Content-Type: auth/request

auth ClueCon
Content-Type: command/reply
Reply-Text: +OK accepted

events plain message
Content-Type: command/reply
Reply-Text: +OK event listener enabled plain

Content-Length: 888

Content-Type: text/event-plain

Event-Name: MESSAGE
Core-UUID: 3ebb22ce-0b58-11e2-9147-f53de47b3be1
FreeSWITCH-Hostname: vz139.gmaruzz.com
FreeSWITCH-Switchname: vz139.gmaruzz.com
FreeSWITCH-IPv4: 192.168.1.139
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2012-10-03%2013%3A51%3A06
Event-Date-GMT: Wed, %2003%20Oct%202012%2011%3A51%3A06%20GMT
Event-Date-Timestamp: 1349265066811330
Event-Calling-File: mod_gsmopen.cpp
Event-Calling-Function: sms_incoming
Event-Calling-Line-Number: 3099
Event-Sequence: 29207
proto: sms
login: gsm01
from: %2B393472665618
date: 10/03/2012%2001%3A51%3A04%20PM%20(%2B0200)
datacodingscheme: default%20alphabet
servicecentreaddress: %2B39391626333
messagetype: 0
subject: SIMPLE%20MESSAGE
to: gsm01
hint: gsm01
to_proto: sms
from_user: %2B393472665618
to_user: gsm01
max_forwards: 70
DP_MATCH: gsm01
skip_global_process: true
dest_proto: GLOBAL
Delivery-Failure: true
Content-Length: 5

Test

gsmopen::dump_event

CUSTOM events of type gsmopen::dump_event are fired in response to a gsmopen_dump command or API call (e.g., from command line or via script or through the ESL). This event reports a lot of useful information on the state of the interface which name was given as argument to the command; if that name is "list" the command will fire as many gsmopen::dump_event as interfaces are running, one for each of them.

This is an example of the gsmopen::dump_event fired in response to:

gsmopen_dump interface2001

Event:
During a call (while a call is active on the interface) a lot of useful info is added (courtesy of Math ;-) 

The command used to generate the event is the same, but the interface is in an active call, executing a JavaScript app:

During a call (while a call is active on the interface) a lot of useful info is added (courtesy of Math ;-) 

The command used to generate the event is the same, but the interface is in an active call, executing a JavaScript app:
Event-Calling-File: mod_gsmopen.cpp
Event-Calling-Function: dump_event_full
Event-Calling-Line-Number: 3008
interface_name: interface2001
interface_id: 1
active: 1
not_registered: 0
home_network_registered: 1
roaming_registered: 0
got_signal: 2
running: 1
imei: 353579017208923
imsi: 222018302196169
controldev_dead: 0
controldevice_name: /dev/ttyACM2
no_sound: 0
alsacname: plughw%3A2
alsapname: plughw%3A2
playback_boost: 1619.086162
capture_boost: 910.479058
dialplan: XML
context: default
destination: 2000
ib_calls: 1
ob_calls: 0
ib_failed_calls: 0
ob_failed_calls: 0
interface_state: 2
phone_callflow: 5
session_uuid_str: 45fbda50-0909-11df-8f99-e9d7ea2264f4
during-call: true
Channel-State: CS_EXECUTE
Channel-State-Number: 4
Channel-Name: gsmopen/interface2001
Unique-ID: 45fbda50-0909-11df-8f99-e9d7ea2264f4
Call-Direction: inbound
Presence-Call-Direction: inbound
Answer-State: answered
Channel-Read-Codec-Name: L16
Channel-Read-Codec-Rate: 8000
Channel-Write-Codec-Name: L16
Channel-Write-Codec-Rate: 8000
Caller-Username: gsmopen
Caller-Caller-ID-Name: GSMopen%3A%20%2B393472665618
Caller-Caller-ID-Number: %2B393472665618
Caller-Destination-Number: 2000
Caller-Unique-ID: 45fbda50-0909-11df-8f99-e9d7ea2264f4
Caller-Source: mod_gsmopen
Caller-Context: default
Caller-Channel-Name: gsmopen/interface2001
Caller-Profile-Index: 1
Caller-Profile-Created-Time: 1264352127793104
Caller-Channel-Created-Time: 1264352127793104
Caller-Channel-Answered-Time: 1264352127793104
Caller-Channel-Progress-Time: 126435212781836
Caller-Channel-Progress-Media-Time: 0
Caller-Channel-Hangup-Time: 0
Caller-Channel-Transfer-Time: 0
Caller-Screen-Bit: true
Caller-Privacy-Hide-Name: false
Caller-Privacy-Hide-Number: false
variable_read_codec: L16
variable_read_rate: 8000
variable_write_codec: L16
variable_write_rate: 8000
variable_channel_name: gsmopen/interface2001
variable_endpoint_disposition: ANSWER
variable_instance_id: 100
variable_current_application_data: freedomfone/leave_message/main.js%20100
variable_current_application: javascript
gsmopen::alarm

CUSTOM events of subtype gsmopen::alarm are automatically fired when something bad happens to an interface, usually resulting in the interface being unavailable for service.

Most interesting fields are:

alarm_code       integer      refers to the type of alarm
alarm_message    string       descriptive text

alarm_code possible values are defined below:

<table>
<thead>
<tr>
<th>GSMopen alarm_code Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

Other fields are the same as in the dump_event event. During a call, the alarm event also gets the additional fields.

This is an example of an alarm event for an interface that fails to initialize at startup (because the physical serial port does not exist):
GSMopen Alarm Event

Content-Length: 1023
Content-Type: text/event-plain

Event-Subclass: gsmopen%3A%3Aalarm
Event-Name: CUSTOM
Core-UUID: 28d9e2e2-068d-11df-8f99-e9d7ea2264f4
FreeSWITCH-Hostname: hardy64
FreeSWITCH-IPv4: 192.168.0.12
FreeSWITCH-IPv6: %3A%3A1
Event-Date-Local: 2010-01-24%2017%3A38%3A15
Event-Date-GMT: Sun,%2024%20Jan%202010%2016%3A38%3A15%20GMT
Event-Date-Timestamp: 1264351095120533
Event-Calling-File: mod_gsmopen.cpp
Event-Calling-Function: dump_event_full
Event-Calling-Line-Number: 3005
alarm_code: 0
alarm_message: gsmopen_serial_init%20failed
interface_name: interface4001
interface_id: 2
active: 0
not_registered: 0
home_network_registered: 0
roaming_registered: 0
got_signal: 0
running: 0
imei: _undef_
imsi: _undef_
controldev_dead: 0
controldevice_name: /dev/ttyACM0
no_sound: 0
alsacname: plughw%3A1
alsapname: plughw%3A1
playback_boost: 10.000000
capture_boost: 5.000000
dialplan: XML
context: default
destination: 2000
ib_calls: 0
ob_calls: 0
ib_failed_calls: 0
ob_failed_calls: 0
interface_state: 0
phone_callflow: 0
session_uuid_str: _undef_
during-call: false

Building

Known Issues

Be sure the dongle has the "voice" capability unlocked, or unlock it with dc-unlocker (http://www.dc-unlocker.com/).

Linux, *BSD, etc

Which Linux distro? Desktop or Server?

Desktop operating systems and distros are completely unsupported.

If you want to use desktop operating systems you have to find the way yourself, sorry.

Only operating systems supported by GSMopen are 64 bit servers: LTS ubuntu 12.04, centos 6, SL6, Debian 6, running directly on the hardware (eg: no Virtual Machines), or in OpenVZ containers running on Debian 6 or Centos 6.
Prerequisites

CentOS 5.x

CentOS 5.x DOES NOT WORK with Huawei and mod_gsmopen, use CentOS 6.x

CentOS 6.x, RHEL6.x, Scientific 6 Server 64-bit

Before building GSMopen module do:

```
<table>
<thead>
<tr>
<th>GSMopen Pre-Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd gsmlib/gsmlib-1.10-patched-13ubuntu</td>
</tr>
<tr>
<td>./configure</td>
</tr>
<tr>
<td>make</td>
</tr>
<tr>
<td>make install</td>
</tr>
<tr>
<td>ldconfig</td>
</tr>
</tbody>
</table>
```

Despite the name of the directory, it works well particularly for CentOS ;)

Check if the library is added by running

```
ldconfig -p | grep gsm
```

If you don't see any records, correct it after reading this post http://linux.101hacks.com/unix/ldconfig/ e.g. adding new rule file in /etc/ld.so.d/. Then re-run ldconfig then:

```
<table>
<thead>
<tr>
<th>Make GSMopen</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd /usr/src/freeswitch/src/mod/endpoints/mod_gsmopen/libctb-0.16/build</td>
</tr>
<tr>
<td>make DEBUG=0 GPIB=0</td>
</tr>
<tr>
<td>make DEBUG=0 GPIB=0 install</td>
</tr>
<tr>
<td>ldconfig</td>
</tr>
</tbody>
</table>
```

Ubuntu LTS 12.04 Server 64-bit

Before building GSMopen module do:

```
<table>
<thead>
<tr>
<th>GSMopen Pre-Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt-get install gsm-utils</td>
</tr>
<tr>
<td>apt-get install libgsmme-dev</td>
</tr>
<tr>
<td>apt-get install usb-modeswitch-data usb-modeswitch</td>
</tr>
</tbody>
</table>
```

then:

```
<table>
<thead>
<tr>
<th>Make GSMopen</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd /usr/src/freeswitch/src/mod/endpoints/mod_gsmopen/libctb-0.16/build</td>
</tr>
<tr>
<td>make DEBUG=0 GPIB=0</td>
</tr>
<tr>
<td>make DEBUG=0 GPIB=0 install</td>
</tr>
<tr>
<td>ldconfig</td>
</tr>
</tbody>
</table>
```

You may need to reboot to have your dongle recognized
Debian 6 (Squeeze) Server 64-bit

Before building GSMopen module do:

### Debian GSMopen Pre-Build

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt-get install gsm-utils</td>
</tr>
<tr>
<td>apt-get install libgsmme-dev</td>
</tr>
<tr>
<td>apt-get install usb-modeswitch-data usb-modeswitch</td>
</tr>
</tbody>
</table>

then:

### Debian Make GSMopen

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd /usr/src/freeswitch/src/mod/endpoints/mod_gsmopen/libctb-0.16/build</td>
</tr>
<tr>
<td>make DEBUG=0 GPIB=0</td>
</tr>
<tr>
<td>make DEBUG=0 GPIB=0 install</td>
</tr>
<tr>
<td>ldconfig</td>
</tr>
</tbody>
</table>

You may need to reboot to have your dongle recognized

### Build and Install

After installing prerequisites (see above), go into mod_gsmopen directory and type:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd /usr/src/freeswitch/src/mod/endpoints/mod_gsmopen/</td>
</tr>
<tr>
<td>make clean</td>
</tr>
<tr>
<td>make install</td>
</tr>
</tbody>
</table>

### Configuration File

Install and edit the gsmopen configuration file:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd /usr/src/freeswitch.trunk/src/mod/endpoints/mod_gsmopen/configs/</td>
</tr>
<tr>
<td>cp gsmopen.conf.xml /usr/local/freeswitch/conf/autoload_configs/</td>
</tr>
<tr>
<td>vi /usr/local/freeswitch/conf/autoload_configs/gsmopen.conf.xml</td>
</tr>
</tbody>
</table>

### Determine the correct device files to use for controldevice_name and controldevice_audio_name

Connect to your usb device /dev/ttyUSBx and then call the modem’s number. If you see “RING” then that device should be the controldevice_name. Usually the highest number is the control device. controldevice_audio_name is usually the device one less than the controldevice_name.

### Start FS and Load GSMopen

Launch FreeSWITCH:

```
/usr/local/freeswitch/bin/freeswitch
```

Then activate debug logging in console and logfile, and load mod_gsmopen:

```
freeswitch@machine> console loglevel 9
freeswitch@machine> fsctl loglevel 9
freeswitch@machine> load mod_gsmopen
```
Windows

GSMopen runs very well on Windows.

GSMopen (mod_gsmopen) is NOT automatically built when you build FreeSWITCH on Windows.

Prerequisites on Windows

Build FreeSWITCH on Windows

You will need the Visual C compiler from Microsoft, commercial version, or the free (as in beer) Visual C Express (requires registration). They both give the same results in our case (eg: no need to buy the commercial version just for GSMopen).

After having downloaded the FS sources from Stash or the packaged FS source release, follow the instructions on how to build FS on Windows. Using Visual C (Express or not):

- Open Freeswitch.sln
- Right-click the main solution node at the top of the Solution Explorer
- Right-click and select Build

This will build FreeSWITCH WITHOUT GSMopen. You must now build the prerequisites (see below) and after prerequisites are built, eventually build mod_gsmopen.

GSMopen on Windows

After having built FreeSWITCH, go into FreeSWITCH source directory, eg: c:/freeswitch, and then go to

```
cd src/mod_gsmopen/gsmlib/gsmlib-1.10-patched-13ubuntu/win32
```

Using Visual C (Express or not):

- Open gsmlib.sln (Note: if you are using 2008 Pro or higher or 2010 Pro or higher this step is not needed; see below)
- Right-click the main solution node at the top of the Solution Explorer
- Right-click and select Build

ERRORS ARE OK!, we're only interested in building the library, and it will be built OK. Errors come from application building; we're not interested in the application here.

Then you must start a Visual Studio Command Prompt (from the "Visual Studio Tools" Start Menu). Inside the Command Prompt window go to

```
c:/freeswitch/src/mod/endpoints/mod_gsmopen/libctb-0.16/build
```

Inside the Command Prompt window execute:

<table>
<thead>
<tr>
<th>GSMopen Visual Studio Command Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>nmake -f makefile.vc DEBUG=1 GPIB=0</td>
</tr>
<tr>
<td>nmake -f makefile.vc DEBUG=0 GPIB=0</td>
</tr>
</tbody>
</table>

Visual Studio Pro Users

If you have Visual Studio 2008 Pro or 2010 Pro you must execute the above nmake command then build mod_gsmopen (gsmlib will be built automatically as a dependency).

Build and Install GSMopen on Windows

Go back to the Visual C compiler from Microsoft, commercial version, or the free (as in beer) Visual C Express (requires registration). They both give the same results in our case (eg: no need to buy the commercial version just for GSMopen).

- Open Freeswitch.sln
- Right-click the main solution node at the top of the Solution Explorer
- Click on "Add" and choose "Existing Project" (Note: If you have VS2008 Pro or higher or VS2010 Pro or higher this step is not needed)
- Navigate to c:/freeswitch/src/mod/endpoints/mod_gsmopen/ and select "mod_gsmopen.2008.vcproj"
- mod_gsmopen is added to the FreeSWITCH solution tree
- Right-click on "mod_gsmopen" and choose "Build"

Configuration File on Windows
Install and edit the gsmopen configuration file:

```
copy c:/freeswitch/src/mod/endpoints/mod_gsmopen/configs/gsmopen.conf.xml c:/freeswitch/Debug/conf

notepad c:/freeswitch/Debug/conf/autoload_configs/gsmopen.conf.xml
```

**Start FS and Load GSMopen on Windows**

Launch FreeSWITCH:
```
c:/freeswitch/Debug/freeswitch.exe
```

Then activate debug logging in console and logfile, and load mod_gsmopen:
```
freeswitch@machine> console loglevel 9
freeswitch@machine> fsctl loglevel 9
freeswitch@machine> load mod_gsmopen
```

**How to Report Bugs and Request Features**

Be sure the **dongle has the "voice" capability unlocked**, or unlock it with dc-unlocker (http://www.dc-unlocker.com/).

You can file bug reports, hints, suggestions, feature requests, improvements, patches, etc to Jira open an account there if you don't have it (it's free ;-) ).

Here’s the best way to give us info on bugs:
1) from the FS CLI: "console loglevel 9"
2) from the FS CLI: "fsctl loglevel 9"
3) from the FS CLI: "unload mod_gsmopen"
4) from the FS CLI: "load mod_gsmopen"
5) reproduce the bug
6) attach the complete, since the beginning* console output (or freeswitch.log file)
   as a file attachment to the Jira bug.
   Please do not cut and paste console output
   or freeswitch.log in the Jira message. ATTACH it.

If the bug involves crashes, core dumps, etc, please read this guide on how to report it Reporting Issues to GitHub, then file a JIRA to mod_gsmopen with all relevant info.

**How to Find Help**

Be sure the **dongle has the "voice" capability unlocked**, or unlock it with dc-unlocker (http://www.dc-unlocker.com/).

You can drop in the IRC channels #freeswitch on irc.freenode.net to ask questions and discuss issues. The original developer Giovanni Maruzzelli of GSMopen is called "gmaruzz" in the IRC channel.

You can also write to the FS users' and developers' mailing lists: http://lists.freeswitch.org/mailman/listinfo but PLEASE do not write to both; choose either IRC or the mailing list to avoid overloading the volunteer users who provide support.

```
! UNLOCK THE DONGLE FIRST

UNLOCK THE DONGLE!!!

Be sure the **dongle has the "voice" capability unlocked**, or unlock it with dc-unlocker (http://www.dc-unlocker.com/).
```

**Troubleshooting**

**Intermittent failures**

MORE THAN ONE DEVICE on a USB bus without a dedicated power supply can have **intermittent failure**, because dongles can use all of the power! Remember, this dongle is a tiny cellular radio and needs plenty of current to transmit. If you use more than one device, use an external (or many, cascaded) power supplies, powered from the wall to supply the USB hub. That's perfectly OK.
VMWare support

Be sure to check your VMWare host supports DirectPath I/O. Check in vCenter under the host's Configuration/Advanced Settings/DirectPath I/O. If it says "Host does not support passthrough configuration" you wont be able to make it work.

See Also

mod_sms