SMSD Configuration File

Description

gammu-smsd reads configuration from a config file. It's location can be specified on command line, otherwise default path `/etc/gammu-smsdrc` is used.

This file use ini file syntax, see *INI file format*.

Configuration file of gammu-smsd consists of at least two sections - `[gammu]` and `[smsd]`. For *SQL Service* you can also use `[sql]`.

The `[gammu]` section is configuration of a phone connection and is same as described in *Gammu Configuration File* with the only exception that `LogFile` is ignored and common logging for gammu library and SMS daemon is used. However the `LogFormat` directive still configures how much messages gammu emits.

`[smsd]`

The `[smsd]` section configures SMS daemon itself, which are described in following subsections. First general parameters of SMS daemon are listed and then specific parameters for storage backends.

`[include_numbers]`

List of numbers from which accept messages, see *Message filtering*.

`[exclude_numbers]`

List of numbers from which reject messages, see *Message filtering*.

`[include_smsc]`

List of SMSC numbers from which accept messages, see *Message filtering*.

`[exclude_smsc]`

List of SMSC numbers from which reject messages, see *Message filtering*.

`[sql]`

Configure SQL queries used by *SQL Service*, you usually don't have to modify them.

See also: *Configurable queries*

General parameters of SMS daemon

`Service`

SMSD service to use, one of following choices:

`FILES`

stores messages in files, see *Files backend* for details

`NULL`

does not store messages at all, see *Null Backend* for details

`SQL`

stores messages in SQL database, see *SQL Service* for details

New in version 1.28.93.

`MYSQL`

synonym for **`Service`** = `SQL` and **`Driver`** = `native_mysql`

stores messages in MySQL database, see *MySQL Backend* for details

Deprecated since version 1.28.93.

`PGSQL`

synonym for **`Service`** = `SQL` and **`Driver`** = `native_pgsql`

stores messages in PostgreSQL database, see *PostgreSQL Backend* for details

Deprecated since version 1.28.93.

`DBI`

synonym for **`Service`** = `SQL` and **`Driver`** = DBI driver

stores messages in any database supported by libdbi, this includes MSSQL, MySQL, PostgreSQL or SQLite databases, see *DBI Backend* for details

Deprecated since version 1.28.93.

> **Note:** Availability of backends depends on platform and compile time configuration.

`PIN`

PIN for SIM card. This is optional, but you should set it if your phone after power on requires PIN.

`NetworkCode`

Network personalisation password. This is optional, but some phones require it after power on.

`PhoneCode`

Phone lock password. This is optional, but some phones require it after power on.

`LogFile`

File where SMSD actions are being logged. You can also use special value `syslog` which will send all messages to syslog daemon. On Windows another special value `eventlog` exists, which will send logs to Windows Event Log.

If you run SMSD as a system daemon (or service), it is recommended to use absolute path to log file as startup directory might be different than you expect.

Default is to provide no logging.

`LogFacility`

Facility to use on logging backends which support it (currently only syslog). One of following chouces:

- `DAEMON` (default)
- `USER`
- `LOCAL0`
- `LOCAL1`
- `LOCAL2`
- `LOCAL3`
- `LOCAL4`
- `LOCAL5`
- `LOCAL6`
- `LOCAL7`

New in version 1.30.91.

`DebugLevel`

Debug level for SMSD. The integer value should be sum of all flags you want to enable.

1

enables basic debugging information

2

enables logging of SQL queries of service backends

4

enables logging of gammu debug information

Generally to get as much debug information as possible, use 255.

Default is 0, what should mean no extra information.

`CommTimeout`

How many seconds should SMSD wait after there is no message in outbox.

Default is 30.

`SendTimeout`

Shows how many seconds SMSD should wait for network answer during sending sms. If nothing happen during this time, sms will be resent.

Default is 30.

`MaxRetries`

How many times will SMSD try to resend message if sending fails.

Default is 1.

`ReceiveFrequency`

The number of seconds between testing for received SMSes, when the phone is busy sending SMSes. Normally a test for received SMSes is done every `CommTimeout` seconds and after each sent SMS.

Default is 0 (not used).

`StatusFrequency`

The number of seconds between refreshing phone status (battery, signal) stored in shared memory and possibly in service backends. Use 0 to disable.

Default is 15.

`LoopSleep`

The number of seconds how long will SMSD sleep before checking for some activity. Please note that setting this to higher value than 1 will have effects to other time based configurations, because they will be effectively rounded to multiply of this value.

Setting this to 0 disables sleeping. Please not this might cause Gammu to consume quite a lot of CPU power.

Default is 1.

`MultipartTimeout`

The number of seconds how long will SMSD wait for all parts of multipart message. If all parts won't arrive in time, parts will be processed as separate messages.

Default is 600 (10 minutes).

`CheckSecurity`

Whether to check if phone wants to enter PIN.

Default is 1 (enabled).

`CheckBattery`

Whether to check phone battery state periodically.

Default is 1 (enabled).

`CheckSignal`

Whether to check signal level periodically.

Default is 1 (enabled).

`ResetFrequency`

The number of seconds between performing a preventive soft reset in order to minimize the cases of hanging phones e.g. Nokia 5110 will sometimes freeze to a state when only after unmounting the battery the phone will be functional again.

Default is 0 (not used).

`HardResetFrequency`

New in version 1.28.92.

> **Warning:** For some phones hard reset means deleting all data in it. Use `ResetFrequency`, unless you know what you are doing.

The number of seconds between performing a preventive hard reset in order to minimize the cases of hanging phones.

Default is 0 (not used).

`DeliveryReport`

Whether delivery reports should be used, one of `no`, `log`, `sms`.

`log`

one line log entry,

`sms`

store in inbox as a received SMS

`no`

no delivery reports

Default is `no`.

`DeliveryReportDelay`

Delay in seconds how long is still delivery report considered valid. This depends on brokeness of your network (delivery report should have same timestamp as sent message). Increase this if delivery reports are not paired with sent messages.

Default is 600 (10 minutes).

`PhoneID`

String with info about phone used for sending/receiving. This can be useful if you want to run several SMS daemons.

When you set PhoneID, all messages (including injected ones) will be marked by this string (stored as SenderID in the database) and it allows more SMS daemons to share a single database.

This option has actually no effect with *Files backend*.

`RunOnReceive`

Executes a program after receiving message.

This parameter is executed through shell, so you might need to escape some special characters and you can include any number of parameters. Additionally parameters with identifiers of received messages are appended to the command line. The identifiers depend on used service backend, typically it is ID of inserted row for database backends or file name for file based backends.

Gammu SMSD waits for the script to terminate. If you make some time consuming there, it will make SMSD not receive new messages. However to limit breakage from this situation, the waiting time is limited to two minutes. After this time SMSD will continue in normal operation and might execute your script again.

The process has available lot of information about received message in environment, check *RunOnReceive Directive* for more details.

`RunOnFailure`

New in version 1.28.93.

Executes a program on failure.

This can be used to proactively react on some failures or to interactively detect failure of sending message.

The program will receive optional parameter, which can currently be either `INIT` (meaning failure during phone initialization) or message ID, which would indicate error while sending the message.

Note: The environment with message (as is in `RunOnReceive`) is not passed to

> the command.

IncludeNumbersFile

File with list of numbers which are accepted by SMSD. The file contains one number per line, blank lines are ignored. The file is read at startup and is reread only when configuration is being reread. See Message filtering for details.

ExcludeNumbersFile

File with list of numbers which are not accepted by SMSD. The file contains one number per line, blank lines are ignored. The file is read at startup and is reread only when configuration is being reread. See Message filtering for details.

IncludeSMSCFile

File with list of SMSC numbers which are accepted by SMSD. The file contains one number per line, blank lines are ignored. The file is read at startup and is reread only when configuration is being reread. See Message filtering for details.

ExcludeSMSCFile

File with list of SMSC numbers which are not accepted by SMSD. The file contains one number per line, blank lines are ignored. The file is read at startup and is reread only when configuration is being reread. See Message filtering for details.

BackendRetries

How many times will SMSD backend retry operation.

The implementation on different backends is different, for database backends it generally means how many times it will try to reconnect to the server.

Default is 10.

Send

New in version 1.28.91.

Whether to enable sending of messages.

Default is True.

Receive

New in version 1.28.91.

Whether to enable receiving of messages.

Default is True.

Database backends options

All DBI, ODBC, MYSQL and PGSQL backends (see *MySQL Backend*, *ODBC Backend*, PostgreSQL Backend, DBI Backend for their documentation) supports same options for configuring connection to a database:

User

User name used for connection to a database.

Password

Password used for connection to a database.

Host

Database server address. It can also contain port or socket path after semicolon, for example `localhost:/path/to/socket` or `192.168.1.1:8000`.

For ODBC this is used as Data source name.

> **Note:** Some database servers differentiate usage of `localhost` (to use local socket) and `127.0.0.1` (to use locat TCP/IP connection). Please make sure your SMSD settings match the database server ones.

New in version 1.28.92.

PC

Synonym for **Host**.

Deprecated since version 1.28.92.

Database

Name of database to use. Please note that you should create tables in this database before using gammu-smsd. SQL files for creating needed tables are included in documentation.

SkipSMSCNumber

When you send sms from some SMS centere you can have delivery reports from other SMSC number. You can set here number of this SMSC used by you and Gammu will not check it’s number during assigning reports to sent SMS.

Driver

SQL driver to use.

Can be either one of native drivers (`odbc`, `native_mysql` or `native_pgsql`) or *DBI Backend* driver.

Depends on what DBI drivers you have installed, DBI supports: `mysql`, `freetds` (provides access to MS SQL Server and Sybase), `pgsql`, `sqlite`, `sqlite3`, `firebird` and `ingres`, `msql` and `oracle` drivers are under development.

SQL

SQL dialect to use. This is specially useful with *ODBC Backend* where SMSD does not know which server it is actually talking to.

Possible values:

- `mysql` - MySQL
- `pgsql` - PostgreSQL
- `sqlite` - SQLite
- `mssql` - Microsoft SQL Server
- `sybase` - Sybase
- `access` - Microsoft Access
- `odbc` - Generic ODBC

New in version 1.28.93.

See also: You can also completely customize SQL queries used as described in *SQL Queries*.

`DriversPath`

Path, where DBI drivers are stored, this usually does not have to be set if you have properly installed drivers.

`DBDir`

Database directory for some (currently only sqlite) DBI drivers. Set here path where sqlite database files are stored.

Files backend options

The FILES backend accepts following configuration options. See *Files backend* for more detailed service backend description. Please note that all path should contain trailing path separator (/ on Unix systems):

`InboxPath`

Where the received SMSes are stored.

Default is current directory.

`OutboxPath`

Where SMSes to be sent should be placed.

Default is current directory.

`SentSMSPath`

Where the transmitted SMSes are placed, if same as `OutboxPath` transmitted messages are deleted.

Default is to delete transmitted messages.

ErrorSMSPath

Where SMSes with error in transmission is placed.

Default is same as `SentSMSPath`.

InboxFormat

The format in which the SMS will be stored: `detail`, `unicode`, `standard`.

`detail`

format used for message backup by *Gammu Utility*, see *SMS Backup Format*.

`unicode`

message text stored in unicode (UTF-16)

`standard`

message text stored in system charset

The `standard` and `unicode` settings do not apply for 8-bit messages, which are always written raw as they are received with extension .bin.

Default is `unicode`.

> **Note:** In `detail` format, all message parts are stored into signle file, for all others each message part is saved separately.

OutboxFormat

The format in which messages created by *gammu-smsd-inject* will be stored, it accepts same values as InboxFormat.

Default is `detail` if Gammu is compiled in with backup functions, `unicode` otherwise.

TransmitFormat

The format for transmitting the SMS: `auto`, `unicode`, `7bit`.

Default is `auto`.

Message filtering

SMSD allows one to process only limited subset of incoming messages. You can define filters for sender number in `[include_numbers]` and `[exclude_numbers]` sections or using `IncludeNumbersFile` and `ExcludeNumbersFile` directives.

If `[include_numbers]` section exists, all values (keys are ignored) from it are used as allowed phone numbers and no other message is processed. On the other side, in [exclude_numbers] you can specify numbers which you want to skip.

Lists from both sources are merged together. If there is any number in include list, only include list is used and only messages in this list are being accepted. If include list is empty, exclude list can be used to ignore messages from some numbers. If both lists are empty, all messages are accepted.

Similar filtering rules can be used for SMSC number filtering, they just use different set of configuration options - [include_smsc] and [exclude_smsc] sections or IncludeSMSCFile and ExcludeSMSCFile directives.

Examples

There is more complete example available in Gammu documentation. Please note that for simplicity following examples do not include [gammu] section, you can look into *Gammu Configuration File* for some examples how it can look like.

Files service

SMSD configuration file for FILES backend could look like:

```
[smsd]
Service = files
PIN = 1234
LogFile = syslog
InboxPath = /var/spool/sms/inbox/
OutboPpath = /var/spool/sms/outbox/
SentSMSPath = /var/spool/sms/sent/
ErrorSMSPath = /var/spool/sms/error/
```

MySQL service

If you want to use MYSQL backend, you will need something like this:

```
[smsd]
Service = sql
Driver = native_mysql
PIN = 1234
LogFile = syslog
User = smsd
Password = smsd
PC = localhost
Database = smsd
```

DBI service using SQLite

For *DBI Backend* backend, in this particular case SQLite:

```
[smsd]
```

```
Service = sql
Driver = sqlite3
DBDir = /var/lib/sqlite3
Database = smsd.db
```

ODBC service using MySQL

For *ODBC Backend* backend, in this particular case using DSN `smsd` server:

```
[smsd]
Service = sql
Driver = odbc
Host = smsd
```

The DSN definition (in `~/.odbc.ini` on UNIX) for using MySQL server would look like:

```
[smsd]
Description         = MySQL
Driver              = MySQL
Server              = 127.0.0.1
Database            = smsd
Port                =
Socket              =
Option              =
Stmt                =

[smsdsuse]
Driver              = MySQL ODBC 3.51.27r695 Driver
DATABASE            = smsd
SERVER              = 127.0.0.1
```

Numbers filtering

Process only messages from 123456 number:

```
[include_numbers]
number1 = 123456
```

Do not process messages from evil number 666:

```
[exclude_numbers]
number1 = 666
```

Debugging

Enabling debugging:

```
[smsd]
debuglevel = 255
logfile = smsd.log
```