OpenSMPPBox svn-r User's Guide

Open Source SMPP proxy

Rene Kluwen
OpenSMPPBox author
Chimit Ltd.

rene.kluwen@chimit.nl

Victor Luchitz
TLV and other patches
Playfon

vluchits@gmail.com

Nikos Balkanas
Documentation and patches
InAccess Networks SA

nbalkanas@gmail.com

OpenSMPPBox svn-r User's Guide: Open Source SMPP proxy

by Rene Kluwen, Victor Luchitz, and Nikos Balkanas

Abstract

This document describes how to install and use OpenSMPPBox, the Open Source SMPP proxy originally developed by Chimit Ltd. and now being developed further by the open source community, namely the Kannel Group.

Revision History

Revision svn-r 2011.12.25

Table of Contents

1. Introduction ..... **??**
- Overview of SMPP ..... **??**
- OpenSMPPBox overview ..... **??**
- Features ..... **??**
- Limitations ..... **??**
- Requirements ..... **??**

2. Installation ..... **??**
- Getting the source code ..... **??**
- Finding the documentation ..... **??**
- Compiling the proxy ..... **??**
- Installing the proxy ..... **??**

3. Using OpenSMPPBox ..... **??**
- Configuring the proxy ..... **??**
  - Configuration file syntax ..... **??**
  - Inclusion of configuration files ..... **??**
  - OpenSMPPBox configuration ..... **??**
  - smpp logins ..... **??**

4. Getting help and reporting bugs ..... **??**

5. Upgrading notes ..... **??**

List of Tables

3-1. opensmppbox Group Variables ..... **??**

Chapter 1. Introduction

This chapter introduces SMPP in general terms, and explains the role of OpenSMPPBox in SMS flow, outlining its duties and features.

Overview of SMPP

The Short Message Peer to Peer (SMPP) protocol is an open, industry standard protocol designed to provide a flexible data communications interface for transfer of short message data between a Message Center, such as a Short Message Service Centre (SMSC), GSM Unstructured Supplementary Services Data (USSD) Server or other type of Message Center and a SMS application system, such as a WAP Proxy Server, EMail Gateway or other Messaging Gateway. It was maintained by the SMS Forum until it reached maturity and was subsequently disbanded in July 2007.

SMPP Release v3.4, its most popular version, launched in 12/9/1999. Now in its latest implementation, v5.0 further development has been discontinued since the disband of the SMS Forum. All protocols and specifications can still be downloaded from http://www.smsforum.net/.

SMPP supports Digital Cellular Network technologies including:

- GSM
- IS-95 (CDMA)
- ANSI-136 (TDMA)
- iDEN

Using the SMPP protocol, an SMS application system called the "External Short Message Entity" (ESME) may initiate an application layer connection with an SMSC over a TCP/IP or X.25 network connection and may then send short messages and receive short messages to and from the SMSC respectively. The ESME may also query, cancel or replace short messages using SMPP.

SMPP supports a full featured set of two-way messaging functions such as:

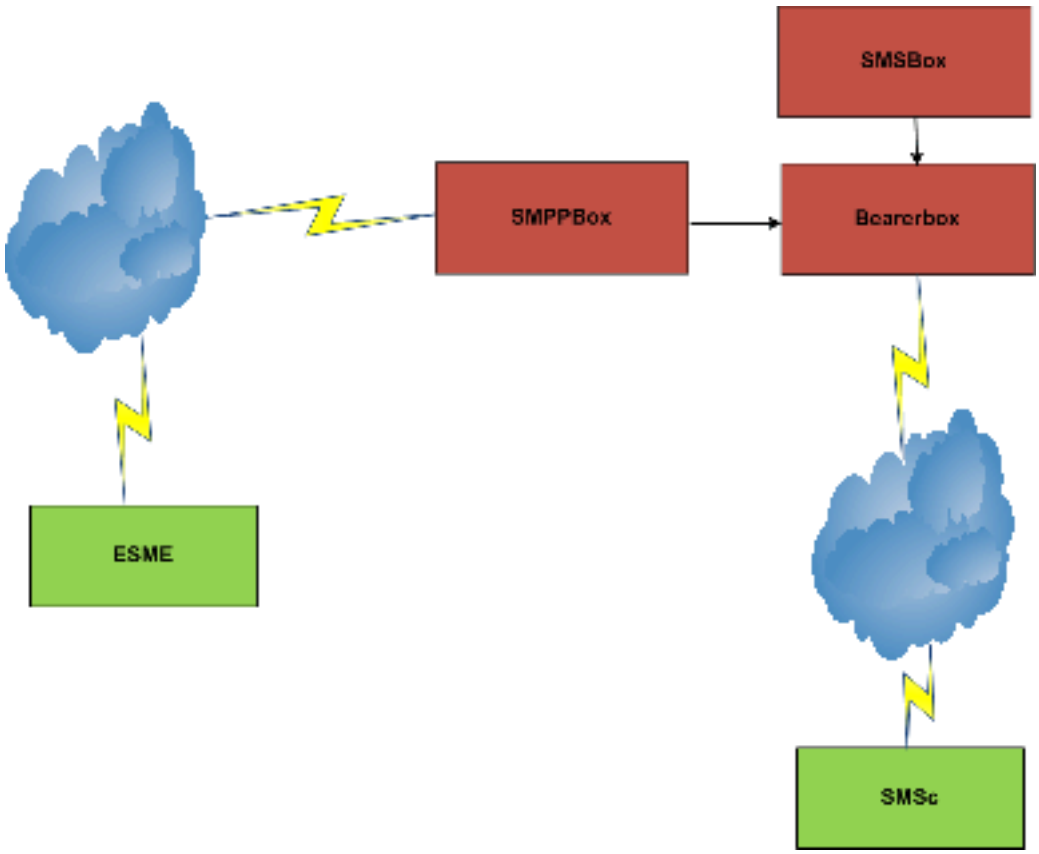
- Transmit messages from an ESME to single or multiple destinations via the SMSC
- An ESME may receive messages via the SMSC from other SME's (e.g. mobile stations).
- Query the status of a short message stored on the SMSC
- Cancel or replace a short message stored on the SMSC
- Send a registered short message (for which a "delivery receipt" will be returned by the SMSC to the message originator)
- Schedule the message delivery date and time
- Select the message mode, i.e. datagram or store and forward
- Set the delivery priority of the short message
- Define the data coding type of the short message

- Set the short message validity period
- Associate a service type with each message e.g. voice mail notification

OpenSMPPBox overview

OpenSMPPBox is an opensource SMPP proxy, which forwards GSM SMPP PDUs. It is not a pure proxy in the clear sense of the word, since it is not limited to the SMPP protocol. It features an SMPP server port for incoming ESME connections, but the client side uses the more flexible Kannel (Msg *) protocol for connection to Kannel's Bearerbox. This way it can take advantage of Bearerbox's client SMSc protocols not limited to SMPP, but extending to CIMD2, EMI/UUCP etc. It can be used for both MT & MO SMS traffic.

Figure 1-1. OpenSMPPBox Layout



The ESME connects over SMPP to OpenSMPPBox, thinking that it is an SMSc. Accounts are configured in OpenSMPPBox to allow connections only from specific clients. The SMS is forwarded to Bearerbox,

which routes it to the best available SMSc over a variety of protocols.

Meanwhile the SMSc will generate both final and intermediate delivery reports. These are routed back from Bearerbox to OpenSMPPbox, which are then rewritten, as to appear that they originated from OpenSMPPBox. These are finally routed back to the requesting ESME.

OpenSMPPBox presents a layer of abstraction to the ESME. The ESME doesn't know the real SMSCs used for SMS delivery. As far as it is concerned, it is dealing only with a single SMSc, OpenSMPPBox. OpenSMPPBox works like a black box in between your subscribers and Kannel.

Features

OpenSMPPBox provides for compliance to SMPP v3.3, SMPP v3.4 & SMPPv5.0 for MT SMS routing over GSM. Options are limited by the features provided by Bearerbox.

SMPP Users are defined in a flat text file, which is parsed at client connection (binding) time. This means that users can be added, changed or removed without restarting opensmppbox. The file can be edited by any plain-text file editor. Also it is possible to compile opensmppbox with Unix PAM support (pluggable authentication modules). See the corresponding options in the configuration file.

It is possible to restrict ip addresses from which can be bound (connected) per user. See the section on configuring opensmppbox below.

Special efforts have been made to make opensmppbox v3.4 compatible by means of TLV (tagged length value) parameters. These parameters can be addressed via the meta-data construction in Kannel. A special example of this: One can conditionally enable transmission of short messages as a whole with length exceeding 140 octets, based on a meta-tag "use_message_payload" in "smpp" group. In case this tag has been set and its value is not zero, opensmppbox will attempt to use the "message_payload" TLV instead of splitting the message into multiple shorter ones with UDH-concatenation bit set. Note that this mechanism only works for ESME's that declare support for SMPP versions 3.4 or greater. A simple usage example:
http://localhost:13013/cgi-bin/sendsms?...&meta-data=%3Fsmpp%3Fuse_message_payload%3D1 ().

Limitations

Some SMPP methods, for instance querying or cancelling short messages are not available.

Billing and logging features are inherited from Kannel, which lacks a great deal of these things. As such, pre-paid billing accounts are not part of the implementation. For post-paid billing, you will need to parse the log-files or possibly use message logging by means of sqlbox.

Requirements

Latest Kannel must be installed (>1.4.3 svn version), including development headers and libraries. Kannel's gwlib is needed for compilation. Additionally a working (running) Bearerbox is needed to route SMS to. If it is not available, SMS messages can possibly be lost and no more logins are permitted.

A C compiler and libraries for ANSI C are needed, with normal Unix extensions such as BSD sockets and related tools. (GNU's GCC tool-chain is recommended)

To build this documentation, the docbook c.s. tools are needed.

Chapter 2. Installation

This chapter explains how the gateway can be installed, either from a source code package or by using a pre-compiled binary version. The goal of this chapter is to get the gateway compiled and all the files in the correct places; the next chapter will explain how the gateway is configured.

Note: If you are upgrading from a previous version, please look at Chapter 5 for any important information. See chapter 5.

Getting the source code

The source code is available from Kannel's site, through svn:

svn co https://svn.kannel.org/opensmppbox/trunk

Authentication is not needed.

Finding the documentation

OpenSMPPBox documentation consists of two parts:

1. *User's Guide*, namely the one you're reading at the moment.
2. The `README`, `ChangeLog` and various other text files in the source tree.

You can also find general information on Kannel's website (http://www.kannel.org) and information about existing problems at our bug tracker (http://bugs.kannel.org).

Everything you need to install and use OpenSMPPBox is in *User's Guide*. The guide is still incomplete in this respect. The `README` is not supposed to be very important, nor contain much information. Instead, it will just point to the other documentation.

Compiling the proxy

If you are using OpenSMPPBox on a supported platform, or one that is similar enough to one, compiling opensmppbox should be trivial. After you have unpacked the source package of your choice, or after you have checked out the source code from SVN, enter the following commands:

```
./configure
make
```

The `configure` script investigates various things on your computer compilation needs, and writes out the `Makefile` used to compile OpenSMPPBox. **make** then runs the commands to actually compile it. It generates the `configure.log`, of all actions taken, usually the first step in debugging in case of errors.

If either command writes out an error message and stops before it finishes its job, you have a problem, and you either need to fix it yourself, if you can, or report the problem to the Kannel project. See Chapter 4 for details.

For detailed instructions on using the configuration script, see file `INSTALL`. That file is a generic documentation for **configure**.

You may need to add compilations flags to configure:

```
CFLAGS='-pthread' ./configure
```

The above, for instance, seems to be required on FreeBSD. If you want to do development, you probably want to add CFLAGS that make your compiler print warning messages. For example, for GCC:

```
CFLAGS='-Wall -g' ./configure
```

(You may, at your preference, use even stricter checking options.)

Installing the proxy

After you have compiled OpenSMPPBox, you need to install certain programs in a suitable place. This is most easily done by using **make** again:

```
make bindir=/path/to/directory install
```

Replace */path/to/directory* with the pathname of the actual directory where the programs should be installed. Actually only a single program is installed `opensmppbox`. The user that runs make install needs to have write permissions do the bindir directory. It defaults to `/usr/local/sbin`. So possibly you need to be root to be able to install. The version number of the proxy is added to the file during installation. This makes it easier to have several versions installed, and makes it easy to go back to an older version if the new version proves problematic.

After installation, you should now be able to run the Kannel init.d script that will start the proxy. Run the script as root. For opensmppbox we don't have a seperate init script, but versions of the Kannel init script are available that include starting opensmppbox.

```
/etc/init.d/kannel start
```

To stop the gateway just run the same script with the stop parameter.

```
/etc/init.d/kannel stop
```

If OpenSMPPBox is already running and you just want to quickly stop and start the gateway,e.g.to set a new configuration option, run the script with the restart parameter.

```
/etc/init.d/kannel restart
```

Chapter 3. Using OpenSMPPBox

This chapter explains how the proxy, OpenSMPPBox, is configured and used. It covers the configuration file and proxy administration during runtime.

There is only one configuration file for all parts of OpenSMPPBox. If several proxy instances are distributed among different hosts, each one needs to have its own configuration file, with its own options.

In bearerbox's status page you can see all connected opensmppbox clients as different smsboxes. Note that the ip address that is listed on the status page of bearerbox is the one of opensmppbox; not the client ip address of the opensmppbox user.

Configuring the proxy

Configuration file syntax

A configuration file consists of groups of configuration variables. Groups are separated by empty lines, and each variable is defined on its own line. Each group in Kannel configuration is distinguished with a group variable. Comments are lines that begin with a number sign (#) and are ignored (they don't, for example, separate groups of variables).

A variable definition line has the name of the variable, and equals sign (=) and the value of the variable. The name of the variable can contain any characters except white space and equals. The value of the variable is a string, with or without quotation marks (") around it. Quotation marks are needed if the variable needs to begin or end with white space or contain special characters. Normal C escape character syntax works inside quotation marks.

Perhaps an example will make things easier to comprehend:

```
1    # Proxy configuration
2    group = opensmppbox
3    bearerbox-host = 127.0.0.1
4    bearerbox-port = 13000
6    opensmppbox-id = smppbox1
7    opensmppbox-port = 13001
8    log-file = /var/log/kannel/opensmppbox.log
9    log-level = 0
10   our-system-id = Inaccess
11   route-to-smsc = fast_smsc
12   # New accounts
13   smpp-logins = /etc/opensmppbox/clients
```

Lines 1 and 12 are comment lines. A blank line is needed to separate groups. The remaining lines define variables. The group type is defined by the group variable value.

The variables used in each configuration group are explained below:

Some variable values are marked as `'bool'`. The value for such a variable is true, false, yes, no, on, off, 0 or 1. Arbitrary values are treated as 'true' while if the variable is missing, it is treated as being 'false'.

In order to make some configuration lines more readable you may use the delimiter '\' at the end of a line to wrap and concatenate the next line up to the current line. Here is an example:

```
1    # A group with a wrapped alias line
2    group = dummy
3    anything = hello
4    aliases = hallo;haalloo;\
5      heelloo;haelloo;healloo
6    whatever = "Hello world!"
```

The above example shows how a list for various alias keywords is wrapped to two lines using the line wrap delimiter. In order to use the delimiter '\' itself, you need to escape it via a prefixed '\' itself. So this is '\\' to escape the wrapping function and use the character in the string.

Inclusion of configuration files

A configuration file may contain a special directive called `include` to include other file or a directory with files to the configuration processing.

This allows to segment the specific configuration groups required for several services and boxes to different files and hence to have more control in larger setups.

Here is an example that illustrates the `include` statement :

```
# OpenSMPPBox configuration

include = "/etc/opensmppbox/conf/opensmppbox1.conf"
```

Above is the main `opensmppbox.conf` configuration file that includes the following `opensmppbox1.conf` file with all required directives for the specific box, and a `configurations` directory which may include more files to include.

```
# opensmppbox1.conf

group = opensmppbox
bearerbox-host = 127.0.0.1
bearerbox-port = 13002
opensmppbox-id = Dutch
opensmppbox-port = 13003
log-file = "/var/log/kannel/opensmppbox.log"
log-level = 1
our-system-id = Inaccess
route-to-smsc = cardboard
smpp-logins = /etc/opensmppbox/clients
```

The above `include` statement may be defined at any point in the configuration file and at any inclusion depth. Hence you can cascade numerous inclusions if necessary. It must be, however, between groups and must contain whole group definitions.

At process start time inclusion of configuration files breaks if either the included file can not be opened and processed or the included file has been processed already in the stack and a recursive loop has been detected.

OpenSMPPBox configuration

OpenSMPPBox configuration *MUST* always include a group for general proxy configuration. This group is named as 'opensmppbox' in configuration file. It doesn't matter if this is the first or a later group in the configuration file.

In it's simplest form, 'opensmppbox' group looks like this:

```
group = opensmppbox
our-system-id = Inaccess
smpp-logins = /etc/opensmppbox/clients
```

Naturally this is not sufficient for any real use. Thus, one or more of the optional configuration variables are used. In following list (as in any other similar lists), all mandatory variables are marked with `(m)`, while conditionally mandatory (variables which must be set in certain cases) are marked with `(c)`.

Table 3-1. opensmppbox Group Variables

Variable	Value	Description
`group (m)`	`opensmppbox`	This is a mandatory variable
`bearerbox-host (o)`	`hostname`	Bearerbox server. FQDN or IP address. Defaults to localhost.
`bearerbox-port (o)`	`port number`	TCP port that bearerbox is listening for incoming opensmppbox connections. Should be the same as smsbox-port configured in bearerbox. Defaults to 13001.
`opensmppbox-id (o)`	`string`	Optional opensmppbox instance identifier. This is used for logging identification.
`opensmppbox-port (o)`	`port number`	TCP port that opensmppbox is listening for incoming ESME connections. Defaults to 2345. If you want a different port number for each client, you will need to run a separate opensmppbox instance for each port you are listening on.

Variable	Value	Description
log-file (o)	filename	Filename that opensmppbox will log messages. If missing, logging is disabled.
log-level (o)	integer (0...5)	Logging level. From maximum (0) to minimum (4). Defaults to 0.
our-system-id (m)	string	Corresponds to SMSC identification transmitted to connected ESMEs.
route-to-smsc (o)	string	Corresponds to smsc-id defined in bearerbox. If set, it will send SMS through this SMSc, else it will let bearerbox route the SMS. Defaults to bearerbox routing.
smpp-logins (m)	filename	File that contains authentication credentials for clients connecting to opensmppbox. This should be a file with a single line per client, with username, password and system-type, seperated by spaces. System-type is a special value. In practice, you should have a different system-type for each connecting client. See description of smpplogins.txt below.
use-systemid-as-smsboxid (o)	boolean	If set to true, this opensmppbox user is authenticating as smsbox to bearerbox as the system-id value (first parameter in smpplogins.txt). If set to false (which is the default) then the smsbox-id is the same as system-type (third parameter in smpplogins.txt). If you are using PAM authentication, then use-systemid-as-smsboxid must be set to true.

Variable	Value	Description
enable-pam (o)	boolean	If set to true, then open smpp will use PAM authentication besides the usual smpplogins.txt file. The smpplogins.txt file takes precedence here. If there the user cannot be found there, opensmppbox will try to use PAM authentication. use-systemid-as-smsboxid must be set to true if enable-pam is also true. For this to work, opensmppbox must be compiled with pam-support (configure --enable-pam).
pam-acl (o)	pam acl account	If enable-pam is true, authentication is done against this pam account. It must be present in /etc/pam.d. If not given, then the value "kannel" is used.
source-addr-ton (o)	number	Manually override source address TON setting for the link. (Defaults to -1, do not override).
source-addr-npi (o)	number	Manually override source address NPI setting for the link. (Defaults to -1, do not override).
source-addr-autodetect (o)	boolean	If defined tries to scan the source address and set TON and NPI settings accordingly. (Defaults to no).
dest-addr-ton (o)	number	Manually override destination address TON setting for the link. (Defaults to -1, do not override).
dest-addr-npi (o)	number	Manually override destination address NPI setting for the link. (Defaults to -1, do not override).
timeout (o)	number	The smpp connection gets dropped if opensmppbox does not receive a valid pdu in this number of seconds. (Defaults to 300).

smpp logins

The smpplogins.txt file, as set by the smpp-logins configuration variable defines all users that are able to bind as ESME to opensmppbox. The first three tokens of this file are the username, password and foreign system-type that form the credentials on which the bind- method of the ESME are being matched with. The last token and defines a source ip address to restrict logins to. An example with two example logins:

```
goodclient    secret  remote *.*.*.*
franchise ourpassword localbox  127.0.0.1;213.110.120.33
```

The first line defines a username ("goodclient"), a password ("secret") and an smsbox-id ("remote"). People can log into this account, originating from any ip address.i The second line defines also a username ("franchise"), password("ourpassword") and an smsbox-id ("localbox"), but besides that there is a restriction on that user. It can only bind from the ip addresses 127.0.0.1 and 213.110.120.33. If ip address(es) is/are given, then only those ip addresses are allowed to connect. It works exactly like connect-allow-ip and connect-deny-ip in Kannel.conf. In that case, connect-deny-ip has a mask of "*.*.*.*".

The third token in de smpp-logins file is the foreign system-type and is important in terms of Kannel's sms routing rules. It is used as smsbox-id when connecting to bearerbox. This means that messages sent via that system-type will get corresponding dlr's back. This also counts for MO messages. Also group = smsbox-route in Kannel.conf "listens" to this value. For this reason, it is important to use a different system-type for each different client unless they should receive each others' messages. In case use-systemid-as-smsboxid = true, then in stead of system-type, system-id will be used as this "smsbox-id" value. You are encouraged to use this feature and set it to true.

Chapter 4. Getting help and reporting bugs

This chapter explains where to find help with problems related to the gateway, and the preferred procedure for reporting bugs and sending corrections to them.

The Kannel development mailing list is users@kannel.org. To subscribe, send mail to users-subscribe@kannel.org. This is currently the best location for asking help and reporting bugs. Please include configuration file and version number.

Chapter 5. Upgrading notes

See the file `UPGRADE` in the source tree.