Propietary protocol for sending and receiving SMS/MMS

LleidaNetworks Serveis Telemàtics, S.L.
devel@lleida.net

21st April 2008

Contents

1 Introduction 5

2 General Commands 6
2.1 login . . . 6
2.2 saldo . . . 6
2.3 ping . . . 6
2.4 infonum . . . 7
2.5 tarifa . . . 7
2.6 quit . . . 7

3 Commands to send single MTs 8
3.1 submit . . . 8
3.2 bsubmit . . . 8
3.3 usubmit . . . 8
3.4 fsubmit, fbsubmit y fusubmit . . . 9
3.5 dsubmit, dbsubmit, y dusubmit . . . 9
3.6 dfsubmit, dbfsubmit y dfusubmit . . . 10
3.7 waplink . . . 10
3.8 dst . . . 11
3.9 msg . . . 11
3.10 filemsg . . . 11
3.11 mmsmsg . . . 12
3.12 envia . . . 13
3.13 acuseon . . . 13
3.14 acuseoff . . . 14
3.15 acuse . . . 15
3.16 acuseack . . . 15
3.17 trans . . . 16

4 SMS reception commands (NO Premium) 18
4.1 allowanswer . . . 18
4.2 incomingmo . . . 18
4.3 incomingmoack . . . 18

5 Commands for receiving SMS (Premium) 20
5.1 deliver . . . 20
5.2 resp . . . 20
5.3 bresp . . . 21
5.4 waplinkresp . . . 21

6 MSIDSN resolution commands **22**

6.1 checkall 22

6.2 rcheckall 22

6.3 checknetwork 23

6.4 rchecknetwork 24

6.5 Possible **rcode** element values 25

1 Introduction

The aim of this protocol is to provide an easy to implement system for IP networks for both sending and receiving SMS or MMS.

All messages exchanged between client and server are line texts ended in return (ASCII 10) and they always contain the same syntax:

```
<lab> <command> [parameters]\n
```

The label is used for easily connect the commands with the generated answers, since the answer is always preceded by the same label as the command. The label has to be an increasing integer sequence.

Example:

```
>> 1 LOGIN user password
<< 1 OK 345 0
>> 2 DST +34666666666
<< 2 OK 1
   ...
```

SMSes can be type MO-MT or MT. The MO-MT ones are SMS paid by the user and are made of a couple of messages consisting of a MO (mobile originated) sent by the user and a confirmation MT (mobile terminated).

The sending of a single MT Push spends credits from the client balance whereas the sending of a reply MT to a MO does not.

The balance for each client is a decimal number being increased whenever a recharge is made and being decreased when sendings are done. The decreasing credit quantity varies depending on the country and the activated options for the MT. You may consult tariffs and countries in our web:

`http://www.lleida.net/red`

The available commands are the following **case insensitive**.

2 General Commands

2.1 login

Command to authenticate to the server.

Syntax:

```
<lab> LOGIN <user> <password>
```

Responses:

```
<lab> OK <credit (integer part)> <credit (decimal part)>
<lab> NOOK <error message>
```

2.2 saldo

Gives back the available MT balance.

Syntax:

```
<lab> SALDO
```

Response:

```
<lab> RSALDO <credit (integer part)> <credit (decimal part)>
```

2.3 ping

This command is sent to the server to check that the client program is up and running.

Syntax:

```
<lab> PING <timestamp>
```

Response (mandatory):

```
<lab> PONG <timestamp>
```

2.4 infonum

This command allows you to get information on a particular mobile phone number. It gives back the code (international code + carrier code), country GSM code and the carrier GSM code (useful information for sending Nokia logos). If any of the above data is unknown its value will be "- -" (two dashes).

Syntax:

```
<lab> INFONUM <number>
```

Response:

```
<lab> RINFONUM <prefix> <country code> <network code>
```

2.5 tarifa

It shows the applicable tariff to a particular number. <*delivery receipt*> and <*sender*> are the service rates that have to be added to the MT price whenever both options are used.

Syntax:

```
<lab> TARIFA <number>
```

Response:

```
<lab> RTARIFA <tariff code> <MT price> <delivery receipt> <sender>
```

2.6 quit

It closes the session.

Syntax:

```
<lab> QUIT
```

Response:

```
<lab> BYE
```

3 Commands to send single MTs

3.1 submit

It sends a MT.

Syntax:

```
<lab> SUBMIT <recipient> <text>
```

Responses:

```
<lab> SUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

3.2 bsubmit

It sends a binary MT.

Syntax:

```
<lab> BSUBMIT <recipient> <data in Base64>
```

Responses:

```
<lab> BSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

3.3 usubmit

It sends an unicode MT.

Syntax:

```
<lab> USUBMIT <recipient> <Unicode text in Base64>
```

Responses:

```
<lab> USUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

3.4 fsubmit, fbsubmit y fusubmit

It sends a MT with dynamic TPOA. By means of this command all the recipients of the MT will receive as the originating address the text/number you have specified.

Syntax:

```
<lab> FSUBMIT <sender> <recipient> <text>
<lab> FBSUBMIT <sender> <recipient> <data in Base64>
<lab> FUSUBMIT <sender> <recipient> <Unicode text in Base64>
```

Responses:

```
<lab> FSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> FBSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> FUSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

3.5 dsubmit, dbsubmit, y dusubmit

Schedules a MT to be sent at a pre-specified day and hour.

Syntax:

```
<lab> DSUBMIT <date> <recipient> <text>
<lab> DBSUBMIT <date> <recipient> <data in Base64>
<lab> DUSUBMIT <date> <recipient> <Unicode text in Base64>
```

The <*date*> parameter is when the server will send the MT and the format should be `YYYYMMDDhhmm[+-]ZZzz`.

The *[+-]ZZzz* part is the timezone (the number of hours to be added or substracted from GMT date). For example, the 1st September 2007 at 13:00 in the Spanish timezone would be 200709011300+0100.

Responses:

```
<lab> DSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> DBSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> DUSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

3.6 dfsubmit, dbfsubmit y dfusubmit

Schedules a MT with dynamic TPOA to be sent at a pre-specified day and hour.

Syntax:

```
<lab> DFSUBMIT <date> <sender> <recipient> <text>
<lab> DFBSUBMIT <date> <sender> <recipient> <datos en Base64>
<lab> DFUSUBMIT <date> <sender> <recipient> <text Unicode en Base64>
```

The <*date*> parameter is when the server will send the MT and the format should be `YYYYMMDDhhmm[+-]ZZzz`.
Responses:

```
<lab> DFSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> DFBSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> DFUSUBMITOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

3.7 waplink

It sends a MT with a web link to a multimedia content.

Syntax:

```
<lab> WAPLINK <recipient> <url> <comment>
```

Notes:

- <*url*>: It is the link to the content to be sent to the mobile It will usually be a link to an image, melody or application.
- <*comment*>: It is the text that comes to the recipient before downloading its contect.

Responses:

```
<lab> WAPLINKOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

3.8 dst

It adds numbers to the recipients list to bulk sending of one message to multiple recipients. Although you may add as many recipient's numbers as you wish it is advisable not to add more than 50 in one go. Should there be more, several *DST* could be concatenated.

Syntax:

```
<lab> DST <number1> <number2> ... <numberN>
```

Responses:

```
<lab> OK <accumulated recipient destinies>
<lab> REJDST <rejected number list>
<lab> NOOK <error>
```

3.9 msg

It sets the text to be sent in a bulk sending.

Syntax:

```
<lab> MSG <text>
```

Responses:

```
<lab> OK
<lab> NOOK <error>
```

3.10 filemsg

It offers the possibility to send files of any type, usually images, polyphonic melodies (MIDI) o real (MP3 or AMR), java programs, etc. It is used as a replacement of the *MSG* command, so the conversation with the server will include several *DST* with the recipient's numbers, a *FILEMSG* and a *ENVIA*.

Syntax:

```
<lab> FILEMSG <mimetype> <data in Base64> <title|text>
```

Notes:

- <*data in Base64*>: Is the file data content encoded in Base64 with neither carriage returns nor line feed.
- <*mimetype*>: Is the type of file. The most common ones are:
  - *image/jpg*: JPEG Image
  - *image/gif*: GIF Image
  - *audio/midi*: MIDI polyphonic ringtone
  - *audio/sp-midi*: SP-MIDI polyphonic ringtone
  - *audio/amr*: AMR real tone
  - *audio/mpeg*: MP3 Real tone
  - *video/3gpp*: 3GP Video
  - *application/java-archive*: JAVA games and apps
  - *application/vnd.symbian.install*: Symbian games and apps
- <*title*|*text*>: Is the page content the user sees before downloading the file. Both title and text should be separated by character "|".

Example:

```
Client <-> Server
<< 3 DST +34666666666 +3460000000
<< 4 FILEMSG image/jpeg AeD...we4EWR News|This morning...
<< 5 ENVIA
```

3.11 mmsmsg

It offers the possibility to send MMS. It is used as a replacement of the *MSG* command, so the conversation with the server will include several *DST* with the recipient's numbers, a *MMSMSG* and a *ENVIA*. This command has two possible syntax:

1. `<lab> MMSMSG <mimetype> <data in Base64> <text>`

   This syntax identical to *FILEMSG*, builds an MMS from the data and the text.

   Example:

```
Client <-> Server
<< 3 DST +34666666666 +34600000000
<< 4 MMSMSG video/3gpp AeD...we4EWR News|This morning...
<< 5 ENVIA
```

2. `<lab> MMSMSG <data in Base64>`

   In this variant, <*data in Base64*> is an MMS without WSP headers, and encoded in Base64.

3.12 envia

It carries out the sending of bulk SMSes with the text set by the last *MSG* to all recipients accumulated through *DST* commands. Once the messages are sent, recipient and text lists are reseted.

Syntax:

```
<lab> ENVIA
```

Responses:

```
<lab> OK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

3.13 acuseon

It establishes the email address where the delivery receipts from all the SMSes sent will arrive, until it is cancelled with the acuseoff command. Whenever the delivery receipt is activated, all SMS sent have an additional cost in terms of credits.

Instead of <*mail*> you can specify the word “INTERNAL” thus the server will send the delivery receipt notifications by protocol instead of sending an email. Therefore the delivery receipt will arrive through the *ACUSE* command originated by the server at the time a new message status is received.

Optional parameters *lang* and *cert_type* are used to to specify that email delivery receipt should also be certified (CERTIFIED SMS service). These optional parameters should be enclosed between square brackets as mentioned in the syntax.

The possible values for *lang* are the following ISO codes:

Code	**Language**
ES	Spanish (default language)
CA	Catalan
EN	English
FR	French
DE	German
IT	Italian
NL	Dutch
PT	Portuguese
PL	Polish
SE	Swedish

The value contained in *cert_type* is always ”D” (Default Service).

Syntax:

```
<etiq> ACUSEON [lang=ES] [cert_type=D] <mail>
```

Notas:

- If CERTIFIED SMS service is activated, delivery receipt notifications can not be sent by protocol (“INTERNAL” word).

Responses:

```
<lab> OK
```

3.14 acuseoff

It cancels a previous *ACUSEON*.

Syntax:

```
<lab> ACUSEOFF
```

Response:

```
<lab> OK
```

3.15 acuse

When a status or delivery receipt notification arrives from a message, sent with this feature and already activated to be received by protocol, it is sent to the client through this command. The server immediately sends the *ACUSE* command when a message status notification is received. If the client is logged out, it will be send once logged in.

When an *ACUSE* command arrives it should be replied with the *ACUSEACKR* command.

Syntax:

```
<lab> ACUSE <id> <recipient> <timestamp delivery receipt> <status>
      <timestamp send> <text>
```

The possible values the <*status*> parameter can take are as follows:

Value	Description
ACKED	Successfully delivered to carrier.
BUFFRED	Switched off or out of coverage.
FAILED	Message could not be delivered to recipient.
DELIVRD	Message delivered to recipient.

3.16 acuseack

Command to confirm the reception of a delivery receipt. If a delivery receipt is not replied with this command, it will be resent to the client each time he logs in to the server.

Syntax:

```
<lab> ACUSEACK <id>
```

The <*id*> parameter has to be the same as the one given by the *ACUSE* command.

Example:

```
Client <-> Server
```

```
>> 2 ACUSEON INTERNAL
<< 2 OK INTERNAL
>> 3 SUBMIT +34600000000 Test
<< 3 SUBMITOK 300 00
>> 5 ACUSEOFF
<< 5 OK
   ...
<< 1 ACUSE 25 +34600000000 1175274575 DELIVRD 1175274565 Test
>> 1 ACUSEACK 25
>> 1 ACUSEACKR
   ...
```

We are sending an "INTERNAL" delivery receipt MT and the message status is received within 10 seconds.

3.17 trans

Start/abort/end a transaction. When in a transaction several submits may be done, but they are not sent until the transaction has finished. In case there were no credit enough to send them all, the transaction is aborted, no message is sent and no credit is spent. If there is enough credit, all messages are sent in one go and the correspondent credit, deducted. There are 3 commands: iniciar, abortar y fin:

iniciar: Starts the transaction.

Syntax:

```
<lab> TRANS INICIAR
```

Responses:

```
<lab> RTRANS INICIAR NOOK <error>
<lab> RTRANS INICIAR OK
```

abortar: Aborts the present transaction.

Syntax:

```
<lab> TRANS ABORTAR
```

Responses:

```
<lab> RTRANS ABORTAR NOOK <error>
<lab> RTRANS ABORTAR OK
```

fin: Ends the transaction and sends all MTs.

Syntax:

```
<lab> TRANS FIN
```

Responses:

```
<lab> RTRANS FIN NOOK <error>
<lab> RTRANS FIN OK <remaining credit (integer)> <credit (decimal)>
```

4 SMS reception commands (NO Premium)

The following commands make possible the reception of MO sent to the allocated virtual long number. It allows you to receive messages in the application without charging a Premiun rate to the final user.

4.1 allowanswer

This command activates/deactivates the option to send MT with the possibility of having response. These MT are sent with your allocated virtual long number as the sender so that users can reply it back. This command has no additional charges.

Syntax:

```
<lab> ALLOWANSWER ON
```

It activates the MT option with allowed reply.

```
<lab> ALLOWANSWER OFF
```

It deactivates the MT option with allowed reply.

4.2 incomingmo

When a MO arrives to the allocated virtual long number, it is sent to the client through this command. The *INCOMIMGMO* command sends it immediately to the server. If the client addressee is logged out it will be sent once logged in.

Syntax:

```
<lab> INCOMINGMO <id> <timestamp> <sender> <recipient> <text>
```

When a *INCOMINGMO* arrives it has to be replied with a *INCOMINGMOACK*.

4.3 incomingmoack

Command to confirm the reception of a MO received in the long number. If a MO is not replied with this command, it will be resent to the client each time he logs in.

Syntax:

```
<lab> INCOMINGMOACK <id>
```

The <id> parameter has to be the same as the one given by the *INCOMINGMO* command.

Example:

```
Client <-> Server
>> 2 ALLOWANSWER ON
<< 2 OK
>> 3 SUBMIT +34600000000 Test
<< 3 SUBMITOK 300 00
>> 4 SUBMIT +34666666666 Test
<< 4 SUBMITOK 299 00
>> 5 ALLOWANSWER OFF
<< 5 OK
   ...
<< 1 INCOMINGMO 34 1119260559 +34600000000 +34649731180 Reply
>> 1 INCOMINGMOACK 34
>> 1 OK
   ...
```

We send two MT with allowed reply and one reply is received.

5 Commands for receiving SMS (Premium)

5.1 deliver

This command is sent by the server when a MO arrives with any of the key words the client has been assigned.

Syntax:

```
<lab> DELIVER <MO id> <timestamp> <sender> <recipient> <text>
```

When a *DELIVER* command arrives, the response has to be a *RESP*, *BRESP* o *WAPLINKRESP* command. It is mandatory to respond all *DELIVER*.

5.2 resp

It sends a response to an incoming MO.

Syntax:

```
<lab> RESP <id MO> <response text>
```

Both the MO id and the *RESP* command label must be the same as those from the *DELIVER* command.

Responses:

```
<lab> NOOK <error>
<lab> ROK <id MO>
```

Example:

```
Client <-> Server
<< 4 DELIVER 12345 1049724310 +34666666666 5015 First text
>> 4 RESP 12345 Reply to first message
<< 4 ROK 12345
<< 5 DELIVER 12346 1049724310 +34666666666 5015 Another text
>> 5 RESP 12346 Reply to second message
<< 5 ROK 12346
```

5.3 bresp

It sends a binary response to an incoming MO.

Syntax:

```
<lab> RESP <id MO> <data in Base64>
```

Both the MO id and the *BRESP* command label must be the same as those from the *DELIVER* command.

Responses:

```
<lab> NOOK <error>
<lab> ROK <id MO>
```

5.4 waplinkresp

It responses to a MO with a WAPLINK.

Syntax:

```
<lab> WAPLINKRESP <id MO> <URL> <comentario>
```

Both the MO id and the *WAPLINKRESP* command label must be the same as those from the *DELIVER* command.

Responses:

```
<lab> NOOK <error>
<lab> RWAPLINKRESPOK <id MO>
```

6 MSIDSN resolution commands

MRS service (MSIDSN Resolver Service) provides the necessary information so that the user could enroute successfully voice, MMS or SMS traffic according to the MCC (Movile Country Code), MNC (Movile Network Code). Additional status data is also available such as MSISDN existence and even mobile subscriber status (on/offline).

The following commands allow you to manage the MSIDSN resolving service.

6.1 checkall

This command requests the MSIDSN total resolution. The server will give back the MCC, MNC, MSISDN existence and even mobile subscriber status (on/offline). All queries done with this command take into account data portability.

The server will originate the *RCHECKALL* command with all the available data on the queried MSIDSN within seconds.

Syntax:

```
<lab> CHECKALL <number>
```

Responses:

```
<lab> CHECKALLOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

6.2 rcheckall

Once the MSIDSN resolution process is completed, the information is delivered through this command. *RCHECKALL* command immediately delivers the information. If the client addressee is logged out it will be sent once logged in.

Syntax:

```
<lab> RCHECKALL <id> <timestamp> <number> <ok> <rcode>
      <mcc> <mnc>
```

Notes:

- <*ok*>: Will be "`1`" if the MRS query has been successfully carried out or "`0`" if an error has occurred.
- <*rcode*>: The current value for this parameter will be the status code or error of the phone number. The possible values of this parameter are specified in the section *"Possible rcode element values"*. (see 6.5)

When a *RCHECKALL* command arrives it has to be replied with the *RCHECKALLACK* command.

Syntax:

```
<lab> RCHECKALLACK <id>
```

The <id> parameter has to be the same as the one given by the *RCHECKALL* command.

Example:

```
Client <-> Server
>> 3 CHECKALL +34600000000
<< 3 CHECKALLOK 300 00
   ...
<< 1 RCHECKALL 34 1119260559 +34600000000 1 0 214 07
>> 1 RCHECKALLACK 34
>> 1 OK
   ...
```

In this example we are querying the resolution of the phone number +34600000000 and the server answers back that the operation has been completed successfully (ok 1), the phone is available (rcode 0) and the carrier is Movistar Spain (214 07).

6.3 checknetwork

This command requests the MSIDSN partial resolution. As a result we just get the MCC and the MNC. All queries done with this command take into account data portability.

The server will originate the *RCHECKNETWORK* command with the MCC and MNC on the queried MSIDSN.

Syntax:

```
<lab> CHECKNETWORK <number>
```

Responses:

```
<lab> CHECKNETWORKOK <remaining credit (integer)> <credit (decimal)>
<lab> NOOK <error>
```

6.4 rchecknetwork

Once the MSIDSN resolution process is completed, the information is delivered through this command. *RCHECKNETWORK* command immediately delivers the information. If the client addressee is logged out it will be sent once logged in.

Syntax:

```
<lab> RCHECKNETWORK <id> <timestamp> <number> <ok> <rcode>
      <mcc> <mnc>
```

Notes:

- *<ok>*: Will be "1" if the MRS query has been successfully carried out or "0" if an error has occurred.
- *<rcode>*: The current value for this parameter will be "0" or negative error code. The possible values of this parameter are specified in the section *"Possible rcode element values"*. (see 6.5)

When a *RCHECKNETWORK* command arrives it has to be replied with the *RCHECKNETWORKACK* command.

Syntax:

```
<lab> RCHECKNETWORKACK <id>
```

The <id> parameter has to be the same as the one given by the *RCHECKNETWORK* command.

Example:

```
Client <-> Server
>> 3 CHECKNETWORK +34600000000
<< 3 CHECKNETWORKOK 300 00
   ...
```

```
<< 1 RCHECKNETWORK 34 1119260559 +34600000000 0 1 214 07
>> 1 RCHECKNETWORKACK 34
>> 1 OK
   ...
```

In this example we are querying the resolution of the phone number +34600000000 and the server answers back that the operation has been completed successfully (ok 1), the query has been completed successfully (rcode 0) and the carrier is Movistar Spain (214 07).

6.5 Possible rcode element values

The status codes allowed by the **rcode** element are the following:

Code	Description
-2	Invalid number to resolve. Invalid specified phone number.
-1	Timeout while processing job or system failure. Timeout period has expired or the service receives from HLR response with "System failure".
0	Successfull operation. The query operation has been completed successfully. / Valid Subscriptor.
1	Invalid query format, data missing or unknown error. A internal format error has occurred, data missing or unknown error has occurred.
2	Timeout while processing job. Timeout period has expired.
3	Subscriber does not exist. Subscriber cannot be identified on the HLR (Home Location Register).
4	Teleservice not provisioned. Subscriber does not support this service.
5	Call barred or CUG reject. When the network operator does not allow to call or send messages to this subscriber or the Subscriber does not pass the check of the CUG (Closer User Group) and was rejected for using Short Message Service.
7	Facility not supported. The Subscriber cannot use the SMS Service currently, the visited network does not allow it. (Example: Roaming).

9	Absent suscriber. Due to mobile phone being switched off or out of coverage.
12	Worng number. Invalid phone number of subscriber.