CONGRATULATIONS

ON

PURCHASING

SMSEAGLE

SMSEAGLE NPE-9300-GPRS/3G

Hardware SMS Gateway

USER'S MANUAL

Document version: 3.32

CONTENTS

1. GET READY TO START

What’s in The BOX

Your SMSEagle box contains:

- SMSEagle Hardware SMS Gateway
- External omnidirectional 3.5dBi GSM antenna (with magnetic foot)
- AC/DC power supply (input voltage: 100-240V)
- Warranty card

Prepare for first start

Your SMSEagle is designed so that you can set it up quickly and start using it right away. Follow the steps below to get started.

Step 1: Connect GSM antenna

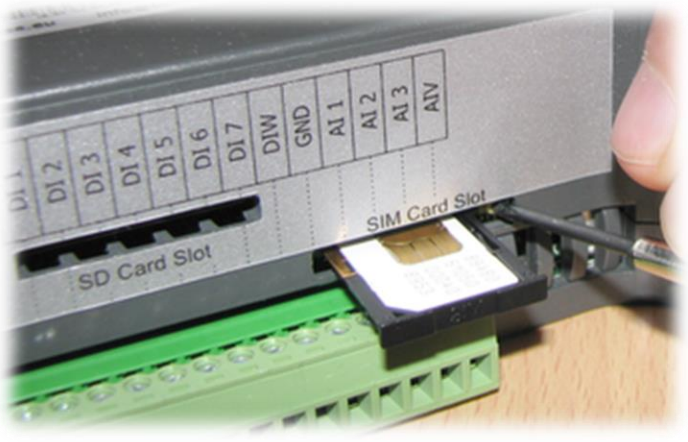
ANTENNA INSTALLATION GUIDELINES

- Install the antenna in a location with access to a cellular network radio signal.
- The antenna must be installed such that it provides a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.
- The antenna must not be installed inside metal cases.

Plug in the antenna connector to the device.

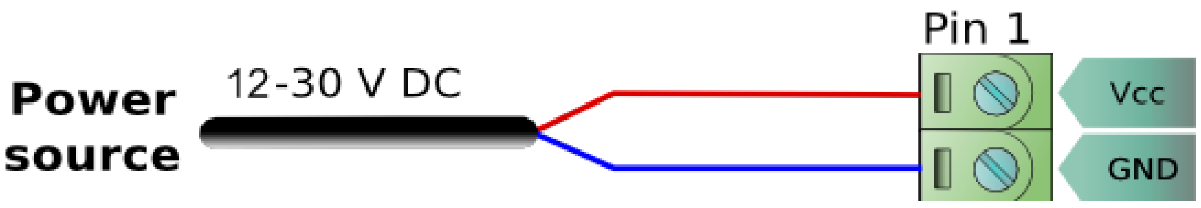
Step 2: Insert SIM Card

Please install SIM Card when the device is SWITCHED OFF. SIM Card slot is located at the bottom of the device. Use a ball-pen or small screwdriver to eject SIM Card tray. Insert card into tray and push it gently into slot.



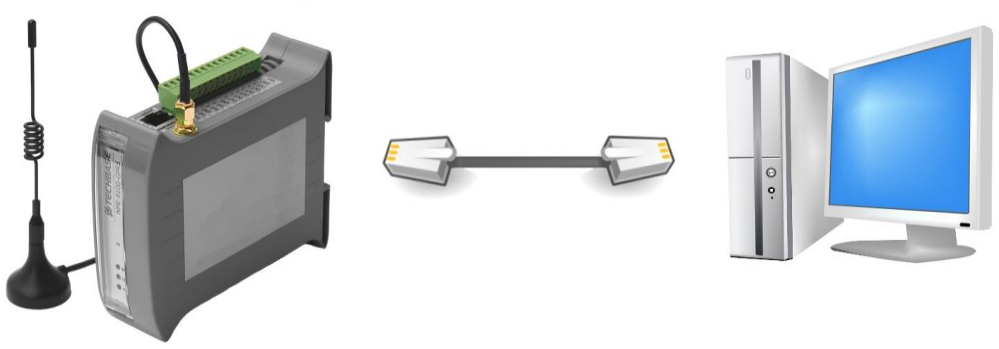
Step 3: Power the device

The device is powered with AC/DC power supply adaptor delivered in the box. The device needs a power source of 12V DC to 30V with 17W power.
In order to power the device simply plug in a connector from AC/DC adaptor into the device.



Step 4: Configure IP settings

Follow the steps below in order to the device first time using Ethernet – with initial factory settings on the device.



SMSEAGLE DEFAULT NETWORK CONFIGURATION

The following configuration is set on a device by default:

IP address: 192.168.0.101

Subnet Mask: 255.255.255.0

DHCP client: ON

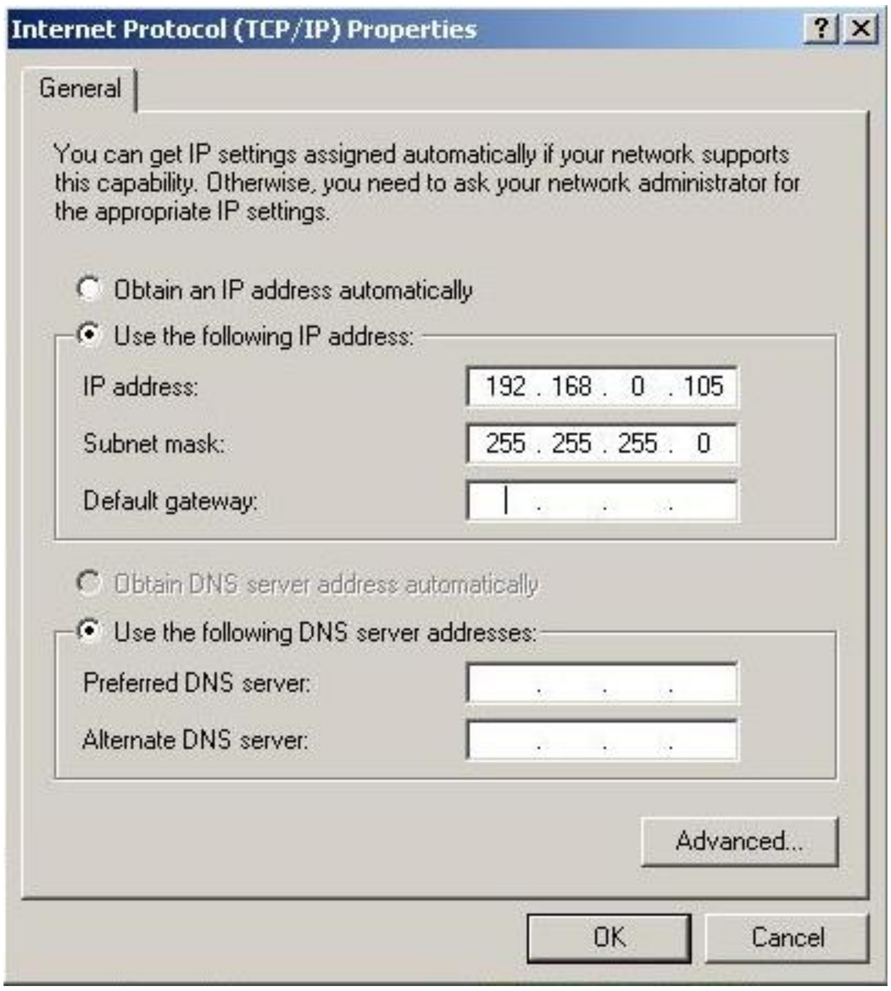
a) CONNECT TO A PC

The computer (PC) must be in the same network subnet as the device. Taking into account the factory settings of the device (listed above), a computer must have following IP configuration:

IP address: 192.168.0.X
Subnet Mask: 255.255.255.0
where X is any value between 1-244 excluding 101
Examples of valid IP addresses are: 192.168.0.1 and 192.168.0.102

The current tutorial assumes you have factory settings on the device. If you do not know what current network settings are you can restore device to the factory settings (see the Troubleshooting chapter).

Example of computer IP configuration in Windows:


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically
(●) Use the following IP address:

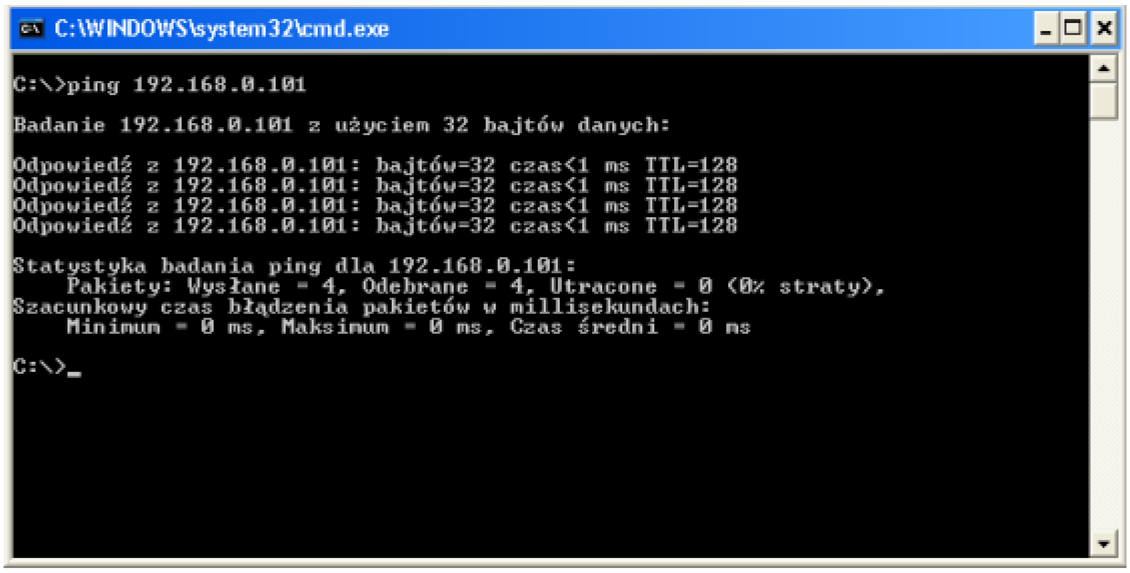
IP address: 192 . 168 . 0 . 105
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

( ) Obtain DNS server address automatically
(●) Use the following DNS server addresses:

Preferred DNS server: . . .
Alternate DNS server: . . .

b) <u>VERIFY YOUR CONNECTION</u>

Properly connected device should respond to the ping command.

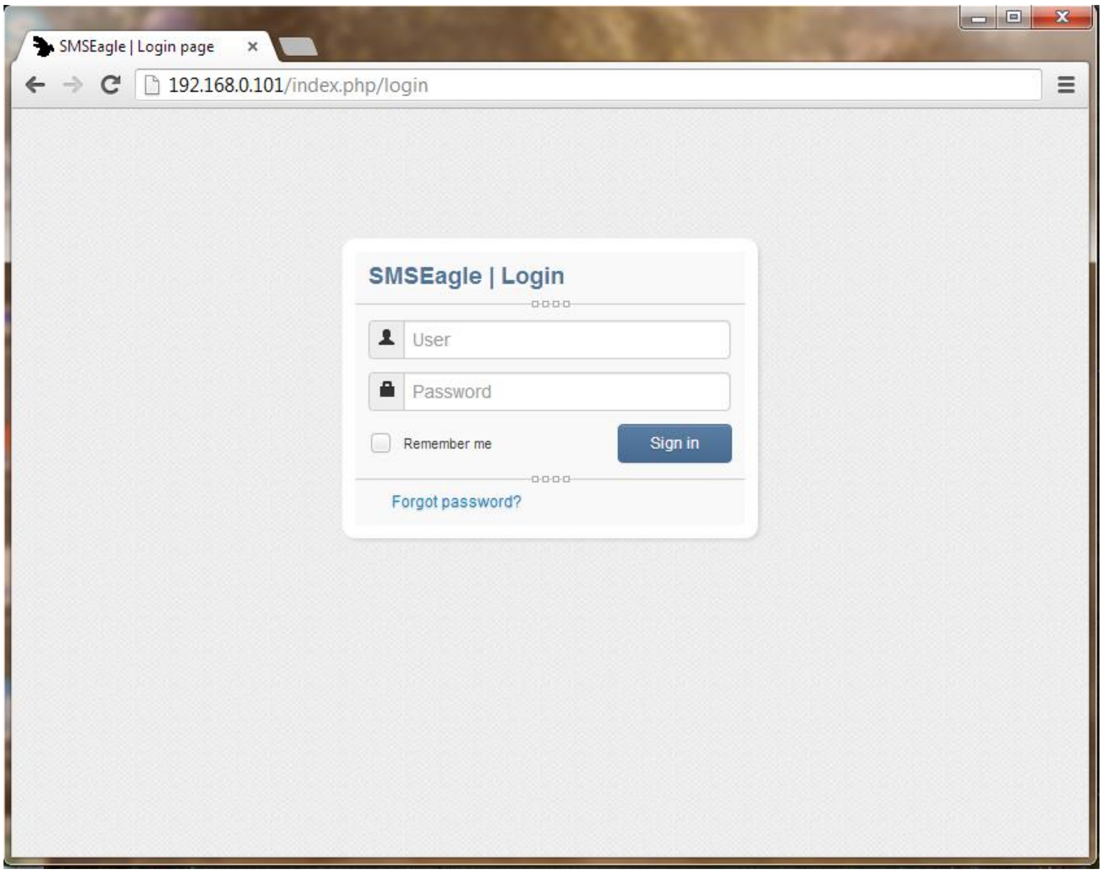

C:\WINDOWS\system32\cmd.exe

```
C:\>ping 192.168.0.101

Badanie 192.168.0.101 z użyciem 32 bajtów danych:

Odpowiedź z 192.168.0.101: bajtów=32 czas<1 ms TTL=128
Odpowiedź z 192.168.0.101: bajtów=32 czas<1 ms TTL=128
Odpowiedź z 192.168.0.101: bajtów=32 czas<1 ms TTL=128
Odpowiedź z 192.168.0.101: bajtów=32 czas<1 ms TTL=128

Statystyka badania ping dla 192.168.0.101:
    Pakiety: Wysłane = 4, Odebrane = 4, Utracone = 0 (0% straty),
Szacunkowy czas błądzenia pakietów w millisekundach:
    Minimum = 0 ms, Maksimum = 0 ms, Czas średni = 0 ms

C:\>_
```

Example of ping after a proper connection to PC

c) <u>LOG IN TO SMSEAGLE</u>

Open an internet browser on your PC and go to the address: 192.168.0.101

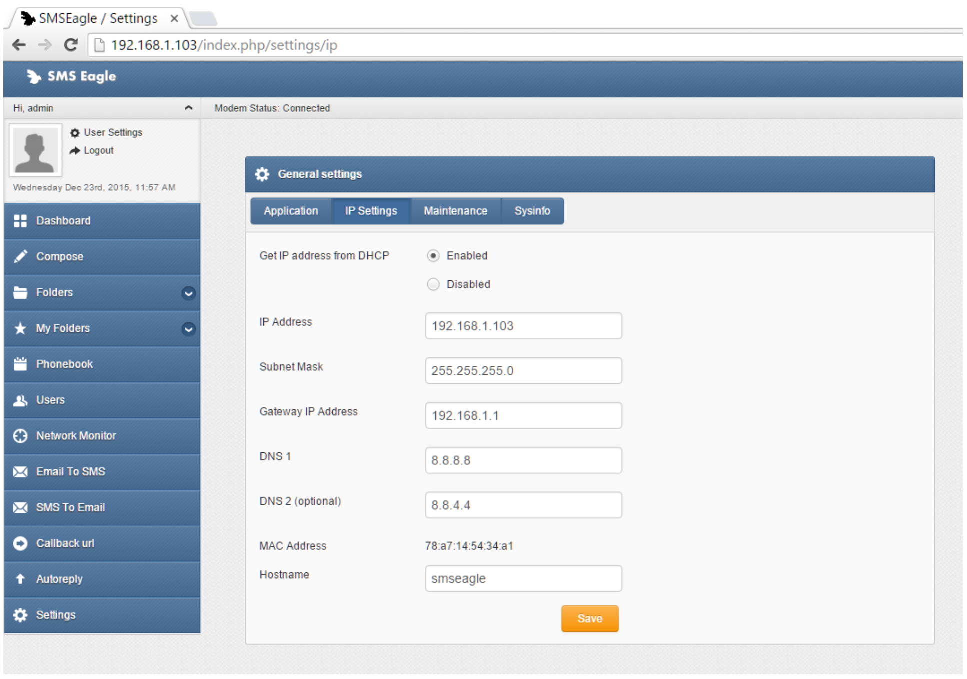


SMSEAGLE DEFAULT USER IS:
Username: admin
Password: password

Login to application with above username and password.

d) Configure IP settings

Click on menu position "Settings" and navigate to tab "IP Settings".



Get IP address from DHCP	[✓ Enabled] Disabled
IP Address	192.168.1.103
Subnet Mask	255.255.255.0
Gateway IP Address	192.168.1.1
DNS 1	8.8.8.8
DNS 2 (optional)	8.8.4.4
MAC Address	78:a7:14:54:34:a1
Hostname	smseagle

Enter your IP settings. If you have DHCP server on your network you can choose “Get IP address from DHCP” – IP settings will be obtained automatically.

Press “Save” button.

e) SETTING SIM-CARD PIN NUMBER

This step should ONLY be done if your SIM-card requires PIN.

If your SIM-card requires PIN number at startup, go to Settings > Maintenance Tab. Enter your PIN number in the field “SIM Card PIN”:

SIM Card PIN

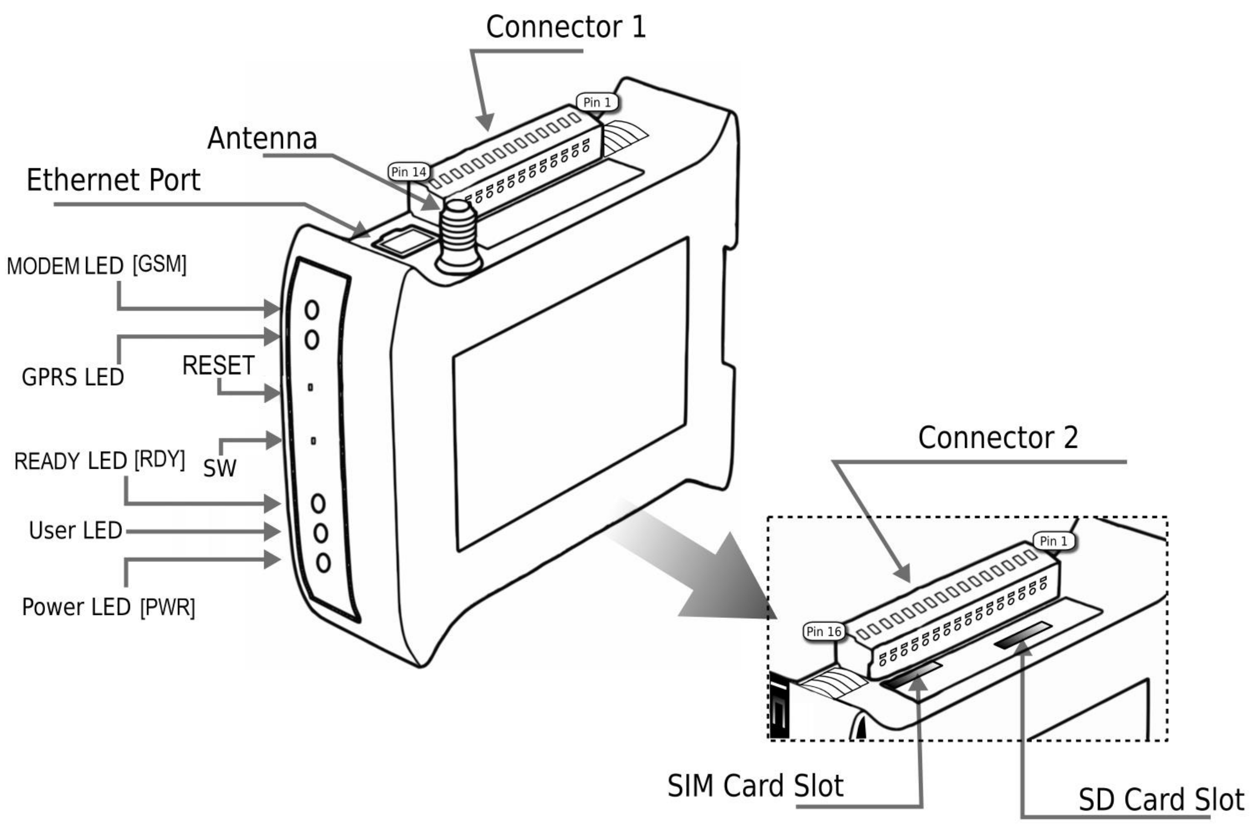
Press “Save” button.

f) REBOOT THE DEVICE

Go to Settings > Maintenance Tab. Press **Reboot** button.

2. USING OF SMSEAGLE

Get to know with Connectors, Ports and LEDs



Element	Label	Description
Connector 1	C1	Power connector and serial ports
Connector 2	C2	Additional ports connector (not used)
SIM Card Slot	SIM	SIM card slot
SD Card Slot	SD	Slot for additional SD/MMC card
Ethernet Port	ETH	Ethernet RJ45 socket
Antenna	ANT	Antenna socket
Power LED	PWR	LED indicating power-on
User LED	USER	LED for user application purpose (not used)
Ready LED	RDY	LED indicating device status
Reset	RST	Switch for rebooting the device
User Switch	SW	Switch for restoring to factory settings

Basic Operations

SMSEagle is capable to work in various screen resolutions, making it accessible for wide range of devices: computers, laptops, tablets, smartphones, etc.

Open a web browser on your device, type in SMSEagle's IP address (as set in previous chapter). At login screen type in your username/password. Default username and password is given in chapter First Start.

SMSEagle basic features

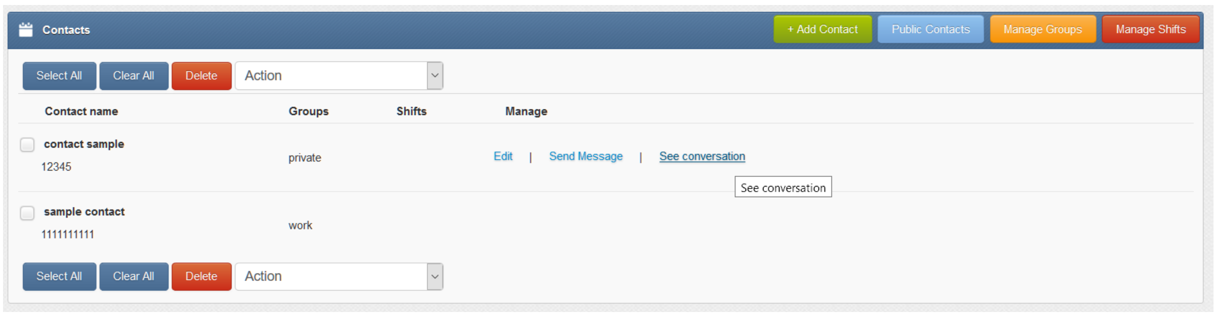
- Sending & Receiving SMS (managing messages with Inbox, Outbox, Sent Items)
- Smartphone-like conversation mode (messages are nicely grouped by phone number). You can easily track history of what you send and receive
- Sending to single numbers, contacts or groups from phonebook
- Import messages for sending from CSV file
- SMS Scheduling by specified date and time or delay
- Message templates (save & edit your own templates)
- Different message types (normal SMS, flash, WAP push, USSD codes)
- Unicode support (support of national characters)
- Multiuser support (each user has access to a private Inbox, Outbox, Sent Items)

Phonebook

Web-GUI of SMSEagle device is equipped with Phonebook for managing contacts, groups and shifts. Each user can create private and public contacts, gather contacts in private and public groups. Contacts can also be optionally assigned to working shifts. Contacts and groups from Phonebook allows users efficient sending of messages.

Phonebook Contacts

Below we present a main Phonebook view, where user manages his Contacts.



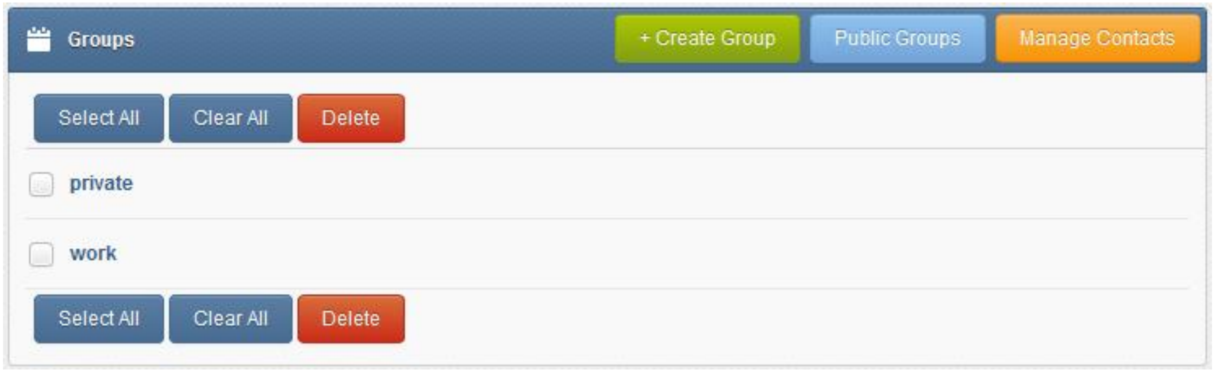
Contact name	Groups	Shifts	Manage
contact sample 12345	private		Edit | Send Message | See conversation
sample contact 1111111111	work		

Screenshot of default phonebook view

In Phonebook Contact Management users can:

- Add/edit/delete contacts via web-gui
- Import contacts from CSV file
- Set contact to public or private visibility
- Add contacts to groups
- Add contacts to working shifts
- Send message to a contact
- View message conversation of a contact

Phonebook Groups



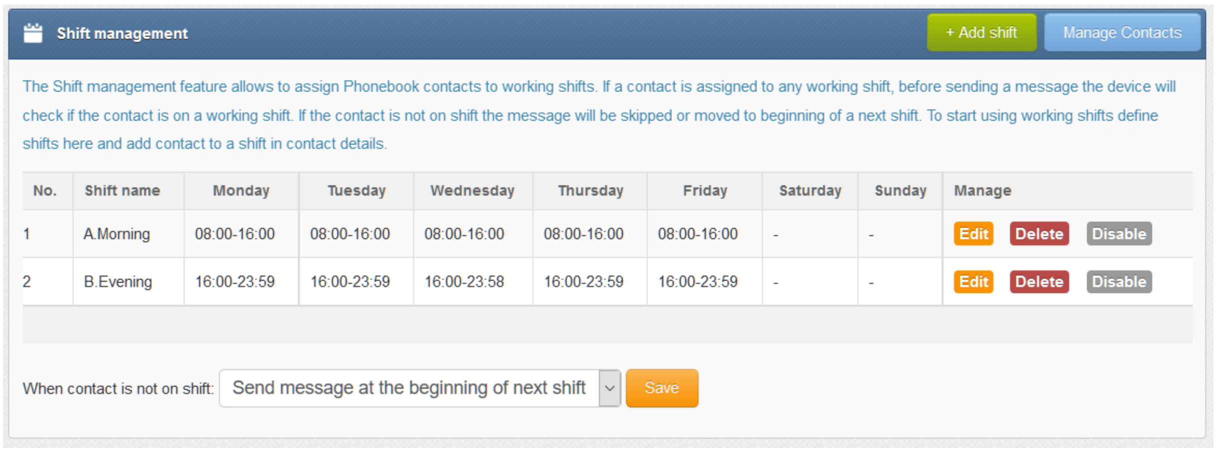
Screenshot taken from phonebook groups

In Phonebook Group Management view users can:

- Add/edit/delete groups
- Set groups to public or private visibility
- View group content (contacts beloning to the group)
- Send message to a group

Phonebook Working Shifts

The Shift management feature allows to assign Phonebook contacts to working shifts. If a contact is assigned to any working shift, before sending a message the device will check if the contact is on a working shift. If the contact is not on shift the message will be skipped or moved to beginning of a next shift. To start using working shifts define shifts here and add contact to a shift in contact details.



The Shift management feature allows to assign Phonebook contacts to working shifts. If a contact is assigned to any working shift, before sending a message the device will check if the contact is on a working shift. If the contact is not on shift the message will be skipped or moved to beginning of a next shift. To start using working shifts define shifts here and add contact to a shift in contact details.

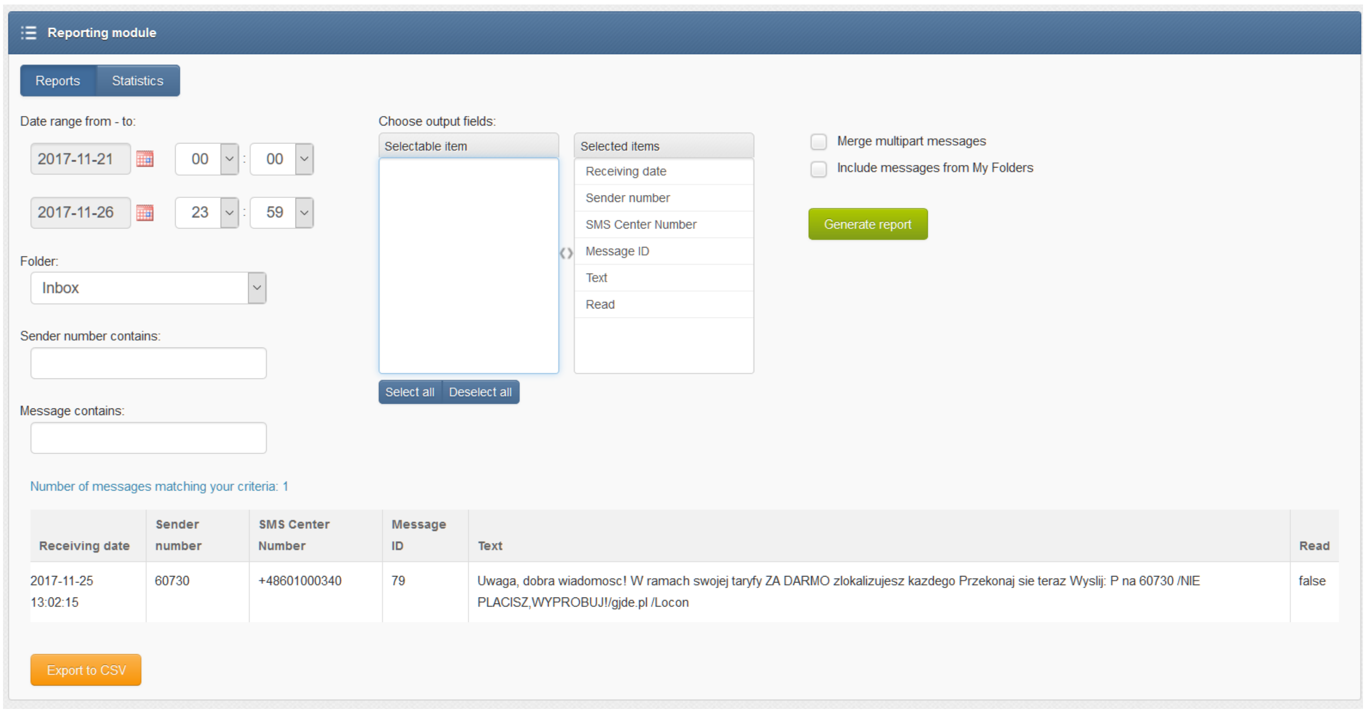
No.	Shift name	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Manage
1	A.Morning	08:00-16:00	08:00-16:00	08:00-16:00	08:00-16:00	08:00-16:00	-	-	Edit Delete Disable
2	B.Evening	16:00-23:59	16:00-23:59	16:00-23:58	16:00-23:59	16:00-23:59	-	-	Edit Delete Disable

When contact is not on shift: Send message at the beginning of next shift

Screenshot of shift management in phonebook

Reporting module

Reporting module is an extension of basic search feature. The module allows users to filter messages from Inbox/Sent items folders based on custom criteria and display filtered messages. Filtered list of messages can be exported to CSV file.



Date range from - to:
2017-11-21 00 : 00
2017-11-26 23 : 59

Folder: Inbox

Sender number contains:

Message contains:

Choose output fields:

Selectable item

Selected items: Receiving date, Sender number, SMS Center Number, Message ID, Text, Read

Merge multipart messages
Include messages from My Folders

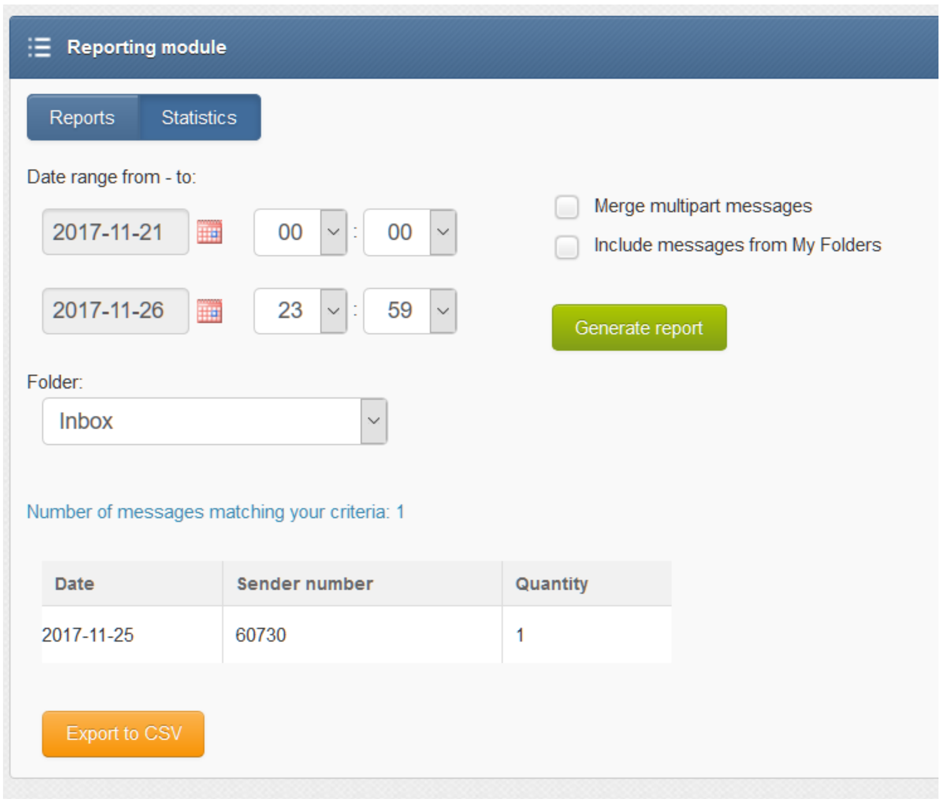
Number of messages matching your criteria: 1

Receiving date	Sender number	SMS Center Number	Message ID	Text	Read
2017-11-25 13:02:15	60730	+48601000340	79	Uwaga, dobra wiadomosc! W ramach swojej taryfy ZA DARMO zlokalizujesz kazdego Przekonaj sie teraz Wyslij: P na 60730 /NIE PLACISZ,WYPROBUJ!/gjde.pl /Locon	false

Screenshot of Reporting module

Statistics view

The reporting module allows also to view daily statistics of sent/received messages. The statistics view displays number of messages per day and sender/receiver number.



Date range from - to:

2017-11-21 00 : 00

2017-11-26 23 : 59

Merge multipart messages

Include messages from My Folders

Folder: Inbox

Number of messages matching your criteria: 1

Date	Sender number	Quantity
2017-11-25	60730	1

Screenshot of Statistics view in Reporting module

SMSEagle plugins

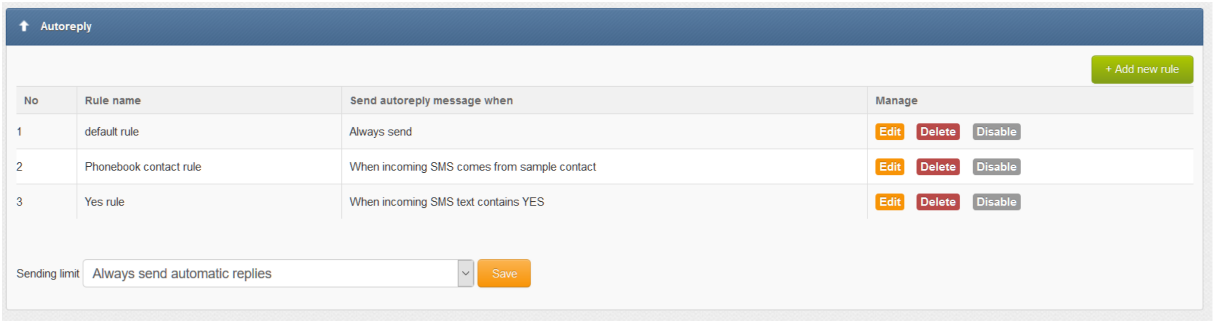
Basic features of SMSEagle software are extended by plugins that provide extra features to the software. Below you will find a description of plugins available in each SMSEagle device. All plugins are an integral part of SMSEagle software. That means that all described plugins are installed in a standard software of SMSEagle device and are available for free.

Autoreply plugin

Plugin allows to automatically respond to each received message with defined text response.

PLUGIN CONFIGURATION

Plugin "Autoreply" allows to add many autoreply rules. Each rule can be enabled or disabled by user.



No	Rule name	Send autoreply message when	Manage
1	default rule	Always send	Edit Delete Disable
2	Phonebook contact rule	When incoming SMS comes from sample contact	Edit Delete Disable
3	Yes rule	When incoming SMS text contains YES	Edit Delete Disable

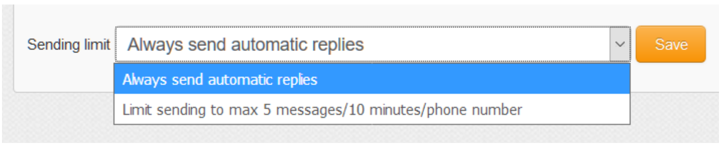
Sending limit: Always send automatic replies

Screenshot from plugin main window

For each rule user can define:

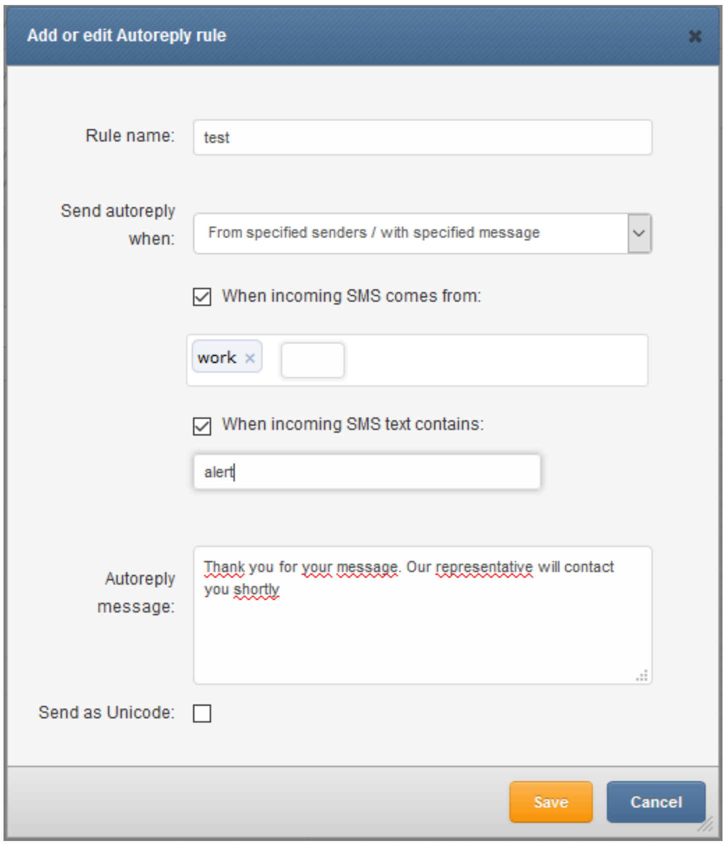
- When autoreply message should be sent:
  - always,
  - when incoming message contains defined text,
  - and/or when message sender belongs to Phonebook contact/group
- If autoreply message text should be sent as Unicode characters

Plugin also allows to define sending limit for autoreply messages. It is possible to set limitation of max 5 messages / 10 minutes / phone number.



Sending limit: Always send automatic replies

Save
Always send automatic replies
Limit sending to max 5 messages/10 minutes/phone number



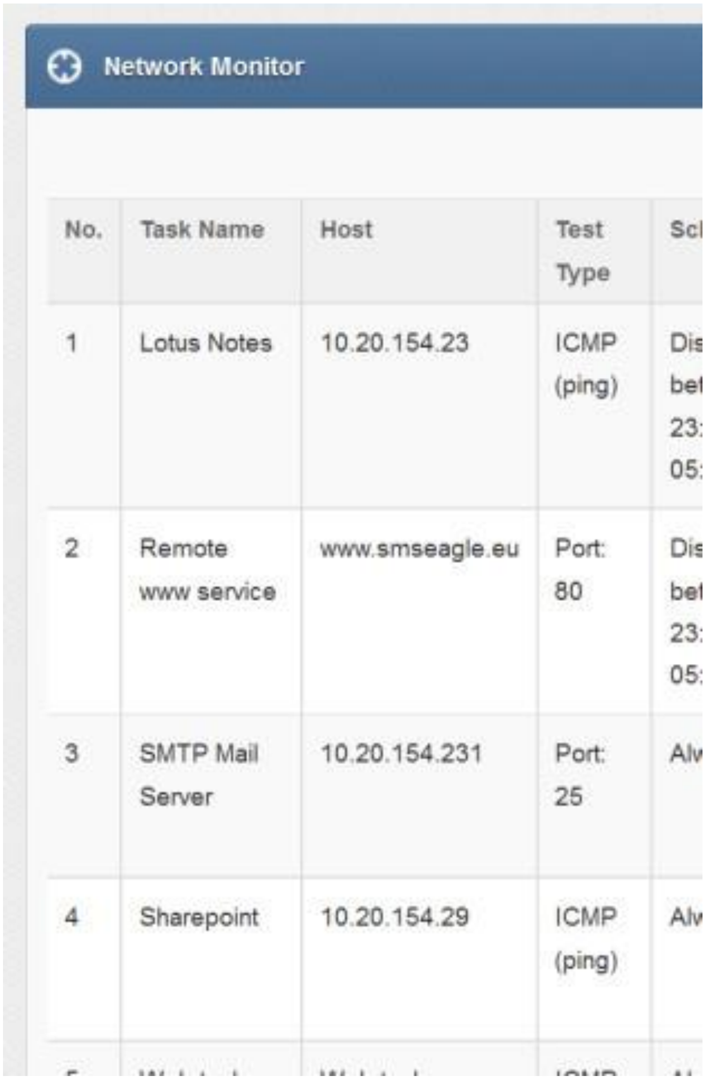
Rule name:	test
Send autoreply when:	From specified senders / with specified message
	[✓] When incoming SMS comes from:
	work ×
	[✓] When incoming SMS text contains:
	alert
Autoreply message:	Thank you for your message. Our representative will contact you shortly
Send as Unicode:	☐

Screenshot form "Add/edit autoreply rule"

Network Monitoring plugin

SMSEagle is equipped with powerful (yet simple to use) network services monitoring features. With that features you can monitor any service that has listening port open. SMSEagle is trying to connect to each defined port in Network Monitoring feature and sends defined SMS alert when port is unavailable. Below you will find a brief overview of plugin capabilities.

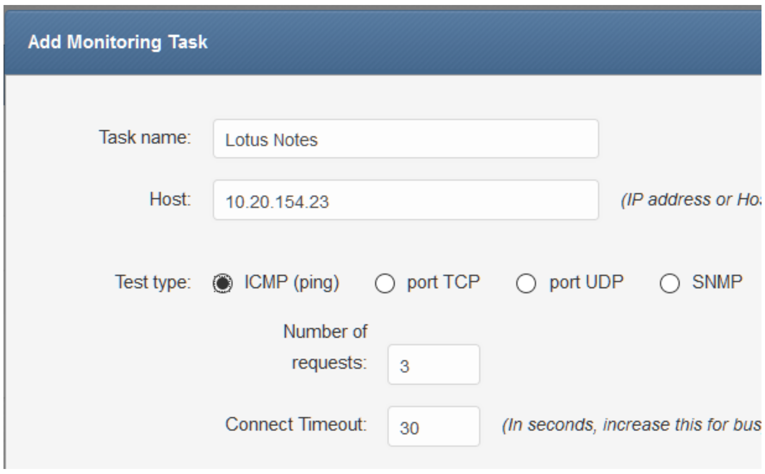
Control status of all your defined tasks


Network Monitor

No.	Task Name	Host	Test Type	Sc
1	Lotus Notes	10.20.154.23	ICMP (ping)	Dis bet 23: 05:
2	Remote www service	www.smseagle.eu	Port: 80	Dis bet 23: 05:
3	SMTP Mail Server	10.20.154.231	Port: 25	Alv
4	Sharepoint	10.20.154.29	ICMP (ping)	Alv

- see a settings' overview for all of your tasks
- check which server/service is currently unavailable
- see when a specific server/service was last down (last downtime)
- check what happened at last downtime (see server/service response)
- edit/delete your tasks
- disable tasks when needed (e.g. when doing a machine upgrades)

Define what you want to monitor in each task


Add Monitoring Task

Task name: Lotus Notes

Host: 10.20.154.23 *(IP address or Ho*

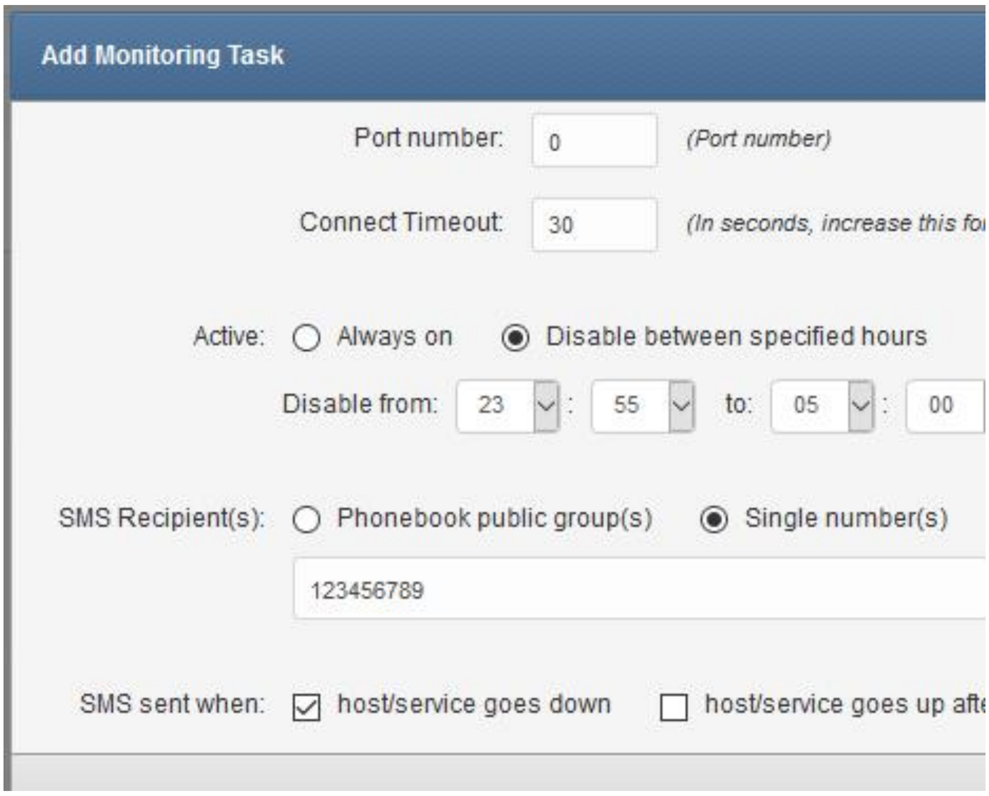
Test type: [✓ ICMP (ping)] port TCP port UDP SNMP

Number of requests: 3

Connect Timeout: 30 *(In seconds, increase this for bus*

- choose a name for the task
- enter a host (IP address or Hostname)
- choose ICMP (ping) to monitor a server with ICMP protocol
- or PORT (TCP/UDP) to monitor your service on a selected port (SMSEagle will check if port is open)
- or SNMP to monitor objects via SNMP protocol (supported return types: numeric, string)
- increase a default timeout value for busy servers (by default we set it to 30 seconds)

Define a schedule


Add Monitoring Task

Port number: 0 (Port number)

Connect Timeout: 30 (In seconds, increase this fo

Active: ○ Always on [● Disable between specified hours]

Disable from: 23 : 55 to: 05 : 00

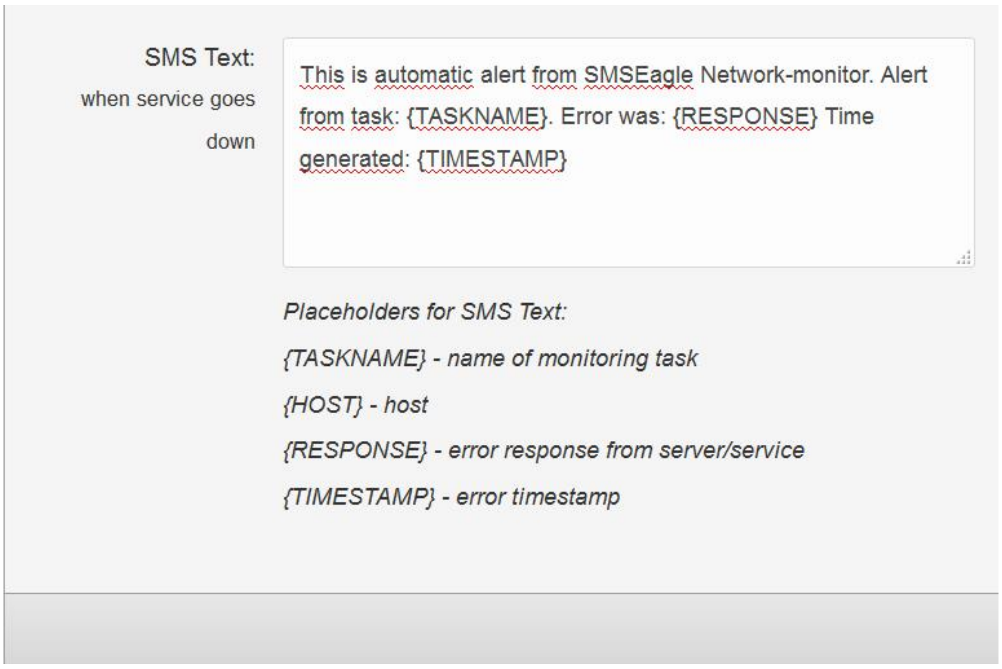
SMS Recipient(s): ○ Phonebook public group(s) [● Single number(s)]

123456789

SMS sent when: [✓ host/service goes down] ☐ host/service goes up afte

- choose if task should be always enabled…
- …or disable it in chosen times
  (during a night, when a machine goes through planned restarts, during resource intensive backups, etc.)
- enter a phone number or choose a group of users to send your SMS alert to
- select when to send SMS alert (when host/service goes down, when host/service goes up after failure)

Define a SMS alert message


SMS Text: when service goes down

This is automatic alert from SMSEagle Network-monitor. Alert from task: {TASKNAME}. Error was: {RESPONSE} Time generated: {TIMESTAMP}

Placeholders for SMS Text:

{TASKNAME} - name of monitoring task

{HOST} - host

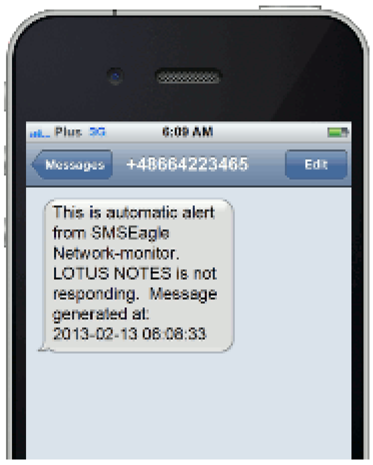
{RESPONSE} - error response from server/service

{TIMESTAMP} - error timestamp

Define your SMS messages when host or service becomes unavailable/comes back to life. Choose field placeholders for your SMS text:

- {TASKNAME} – puts a taskname inside SMS text
- {HOST} – hostname or IP address
- {RESPONSE} – message received (in case of no response from server/service)
- {TIMESTAMP} – timestamp of an error

Receive SMS alerts



This is automatic alert from SMSEagle Network-monitor. LOTUS NOTES is not responding. Message generated at: 2013-02-13 06:08:33

- be immediately alerted when your services/servers go down (or go up after failure)
- give yourself a chance to react quickly

Go to our website www.smseagle.eu for more details of this plugin.

Email to SMS plugin

Email To SMS plugin allows you to convert an email to SMS message.

BASIC USAGE

If the plugin is enabled, email sent to the email address:
PHONE_NUMBER@[IP_ADDRESS_OF_SMSEAGLE] will be converted to SMS message.

PHONE_NUMBER is a destination phone number
IP_ADDRESS_OF_SMSEAGLE is the IP address of your device.
The text of the email is the text of the SMS message (optionally you can append email subject at the beginning of SMS message).

Example: email message sent to the address: 123456789@[192.168.0.101] will be converted to SMS message and delivered to phone number 123456789.

SEND TO USERNAME/GROUP

Email sent to the email address:
NAME_IN_PHONEBOOK@[IP_ADDRESS_OF_SMSEAGLE] will be converted to SMS message and will be sent to a user or users group from SMSEagle's phonebook.

NAME_IN_PHONEBOOK is a username or group name (must be a public group) from SMSEagle's phonebook
IP_ADDRESS_OF_SMSEAGLE is the IP address of your device.
The text of the email is the text of the SMS message (optionally you can append email subject at the beginning of SMS message).

Example: email message sent to the address: db-admins@[192.168.0.101] will be converted to SMS message and delivered to all members of db-admin group. The db-admin group must be defined in your SMSEagle phonebook.

USING FQDN IN EMAIL ADDRESS

It is also possible to use Fully Qualified Domain Name in an email address sent to SMSEagle box (eg.: 123456789@mydomain.com). Please refer to our FAQ article: How do I configure Email2SMS plugin to accept FQDN email addresses for more details.

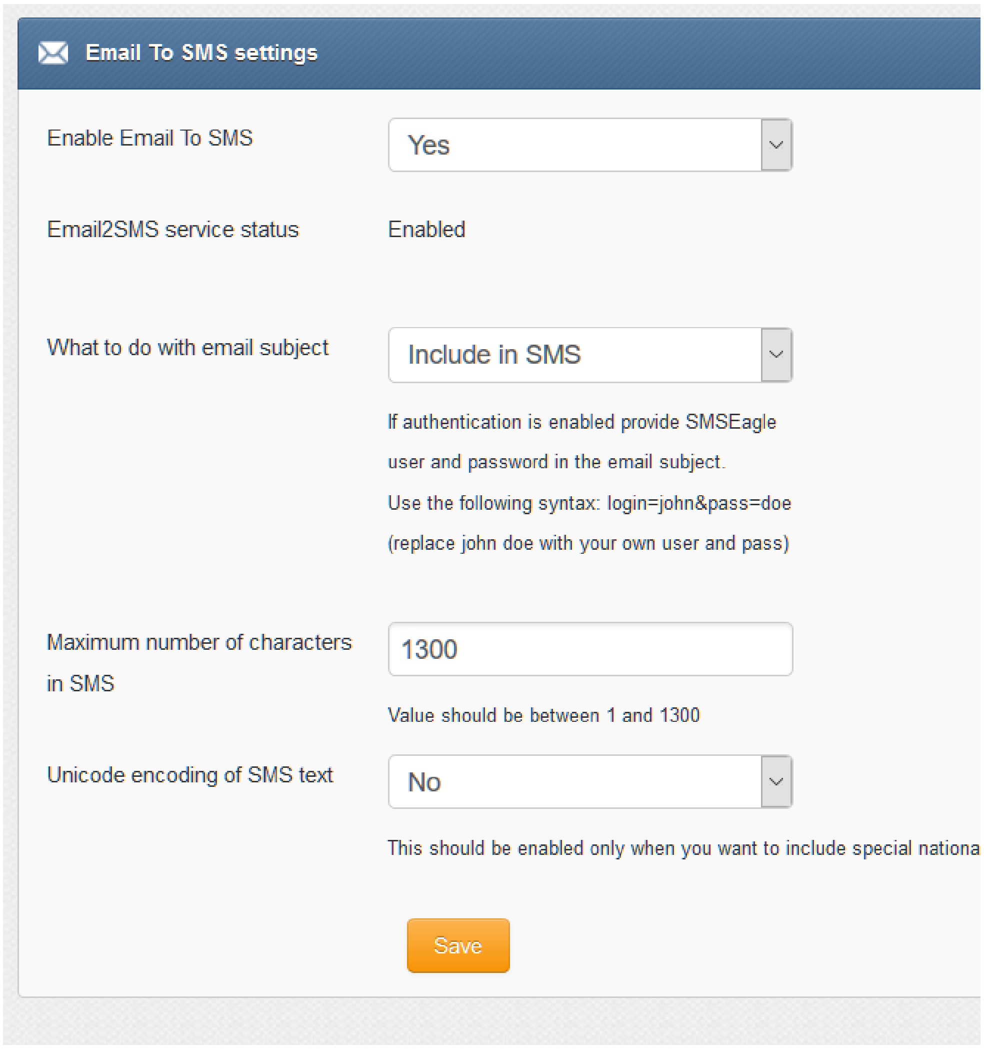
EMAIL SUBJECT- ADDITIONAL PARAMETERS (OPTIONAL)

It is possible to set additional flags for single converted message using email subject. Currently the following flags are available:

- date - date and time in format YYYYmmDDHHMM (YYYY – year, mm – month, DD – day, HH – hour, MM – minute). If this parameter is not null SMS will be scheduled for sending at the given date and time
- modem_no - sets sending modem number (available only for multimodem devices)

If you send email with subject containing FLAG=VALUE the flag will be set for this particular email2SMS message.

*Example 1: email message with subject containing **modem_no=2** will be converted to SMS message and sent via modem number 2.*
*Example 2: email message with subject containing **date=201801010005&modem_no=2** will be converted to SMS message and sent on 2018-01-01 00:05 via modem number 2.*


Email To SMS settings

Enable Email To SMS	Yes
Email2SMS service status	Enabled
What to do with email subject	Include in SMS
	If authentication is enabled provide SMSEagle user and password in the email subject. Use the following syntax: login=john&pass=doe (replace john doe with your own user and pass)
Maximum number of characters in SMS	1300
	Value should be between 1 and 1300
Unicode encoding of SMS text	No
	This should be enabled only when you want to include special nationa

Screenshot from Email to SMS settings

- if you want to use the plugin, set ‘Email2sms active’ to ‘Yes’
- if you want to include a subject of an email in SMS message, set ‘What to do with email subject’ setting to ‘Include in SMS’. The email subject will be appended at the beginning of SMS message
- if you want to use user authentication, set ‘What to do with email subject’ setting to ‘Use for authentication’. If user authentication is enabled, provide in a subject of an email your login and password in the following form: login=john&pass=doe
- if you want to include only a subject of an email in SMS message, set ‘What to do with email subject’ setting to ‘Send only subject without email body’. Only the email subject will inserted in SMS message

- the text of an email will be cropped to the value 'Maximum number of characters'. Maximum allowed length of SMS message is 1300 characters
- if you want to include in SMS message special national characters (like ąäàöß 我) set "Unicode encoding of SMS text" to 'Yes'

Email to SMS Poller

Email2SMS Poller is an alternative for Email2SMS plugin for convert incoming emails to SMS message. This plugin should be used when you need to fetch emails from existing mailbox on your mail server. The Email2SMS Poller plugin connects to configured email account and polls it in specified periods of time for new emails. Once new email is received, it is automatically converted to SMS message.

The plugin supports POP3 and IMAP accounts.

To send SMS using Email2SMS Poller you have to send an email to specified email account, with email subject containing mobile number or phonebook contact/group name.

BASIC EXAMPLE

For example, such email message:

TO: smseagle@mycompany.com
FROM: john.doe@mycompany.com
SUBJECT: +48333444555
BODY: Hello world!

In this case SMSEagle gateway will fetch incoming email from smseagle@mycompany.com account and send it's body as SMS message to +48333444555 mobile number.

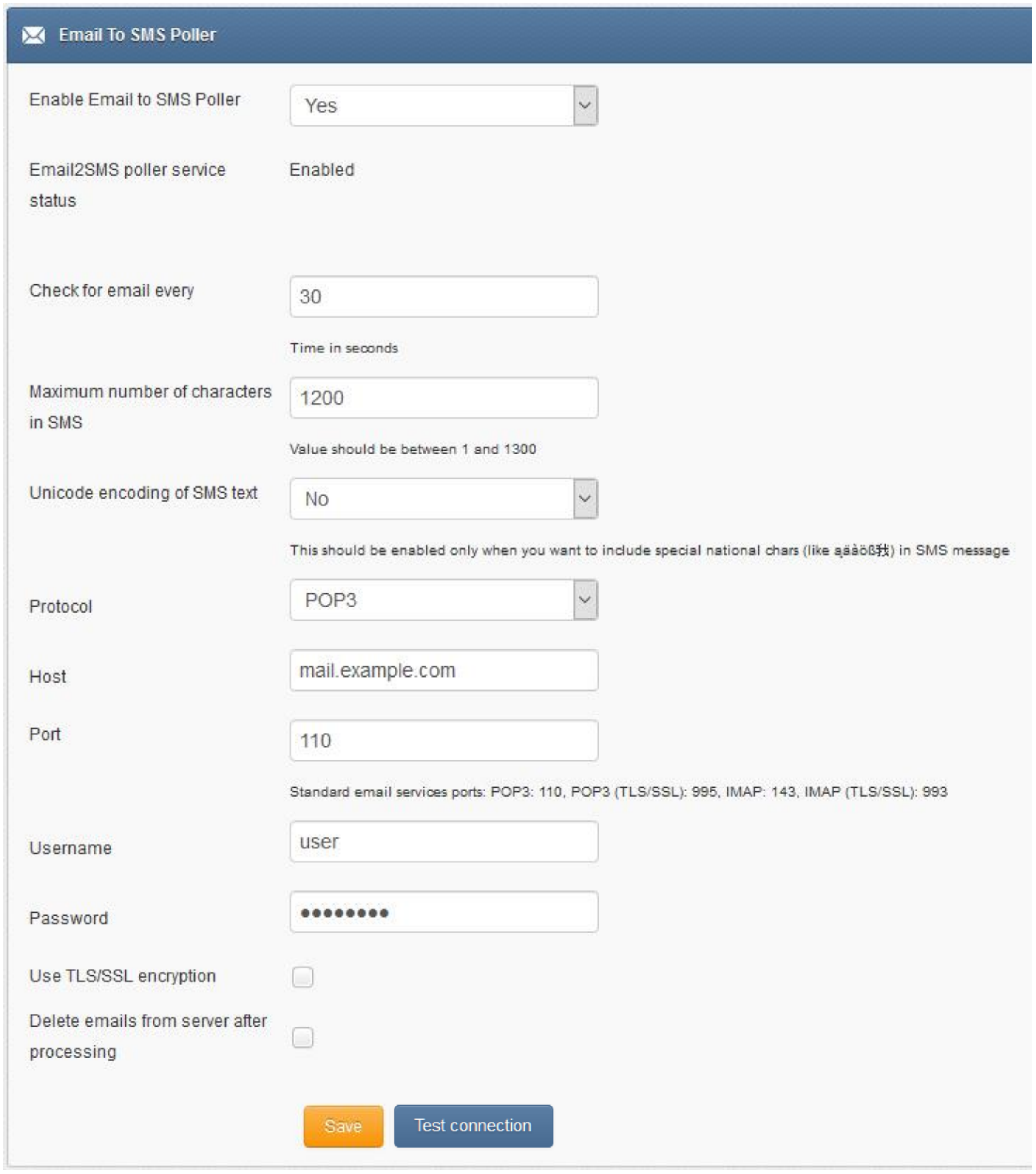
SEND TO USERNAME/GROUP

If you want to send SMS to a contact or group from SMSEagle phonebook, put the contact/group name in SUBJECT field.

Notice:

Messages that are processed by Email2SMS Poller (but not deleted) are marked in the mailbox as read. Software is based on flagging messages- Read/Unread. Marking a read message in the mailbox as unread will result in being processed again by Email2SMS Poller. We suggest using a separate email account to avoid situation with resending the same message (marking unread already processed read message).

PLUGIN CONFIGURATION



Email To SMS Poller

Enable Email to SMS Poller	Yes
Email2SMS poller service status	Enabled
Check for email every	30 Time in seconds
Maximum number of characters in SMS	1200 Value should be between 1 and 1300
Unicode encoding of SMS text	No This should be enabled only when you want to include special national chars (like ąäàöß我) in SMS message
Protocol	POP3
Host	mail.example.com
Port	110 Standard email services ports: POP3: 110, POP3 (TLS/SSL): 995, IMAP: 143, IMAP (TLS/SSL): 993
Username	user
Password	●●●●●●●●
Use TLS/SSL encryption	☐
Delete emails from server after processing	☐

Screenshot from Email to SMS Poller

- if you want to use the plugin, set ‘Enable Email2sms Poller’ to ‘Yes’
- Set email fetching interval (in seconds)
- the text of an email will be cropped to the value ‘Maximum number of characters’. Maximum allowed length of SMS message is 1300 characters.
- If you want to include special national characters, enable "Unicode encoding of SMS text"
- Choose protocol from IMAP or POP3
- Provide mailbox configuration (host, port, user, password, encryption settings)
- If you want to delete emails from the mailbox after they are fetched by Email2SMS Poller, please mark "Delete emails from server after processing"

SMS to Email plugin

SMS to Email plugin allows you to forward your SMS messages to email address.

The plugin can be used in two modes:

a. forwarding of incoming SMS to email of last sender (so called **Two-way Email2SMS & SMS2Email**)
In this mode, when SMSEagle receives incoming SMS, it checks if earlier anyone was sending SMS to the number from incoming SMS using Emai2SMS. If last sender is found, the incoming SMS is forwarded to the email address of last sender. If no last sender is found, then the incoming message is forwarded to a default email address given in plugin settings.

b. It forwards all the incoming messages to one fixed email address.
In this mode all incoming SMS messages are forwarded to always the same email address.

Plugin uses an external SMTP server for sending emails.

EMAIL TEXT FROM PLUGIN

Email body from SMS To Email plugin contains:

- phone number from incoming SMS (and phonebook contact name if found)
- Date, time when SMS is received
- SMS message

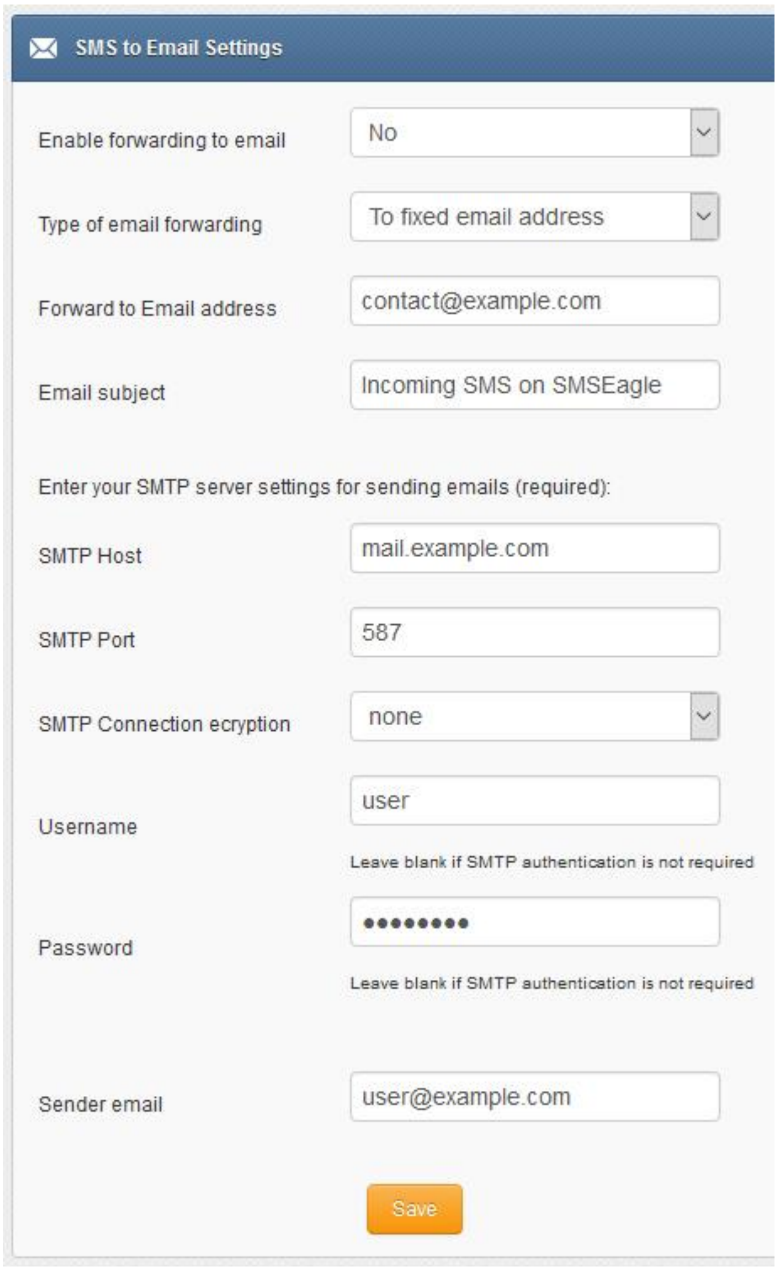
Example email text:

From: +483334455 (John Doe)

Received: 2017-06-01 14:38:12

Message: My SMS message

PLUGIN CONFIGURATION


SMS to Email Settings

Enable forwarding to email	No
Type of email forwarding	To fixed email address
Forward to Email address	contact@example.com
Email subject	Incoming SMS on SMSEagle

Enter your SMTP server settings for sending emails (required):

SMTP Host	mail.example.com
SMTP Port	587
SMTP Connection ecryption	none
Username	user Leave blank if SMTP authentication is not required
Password	●●●●●●●● Leave blank if SMTP authentication is not required
Sender email	user@example.com

Screenshot from SMS to Email settings

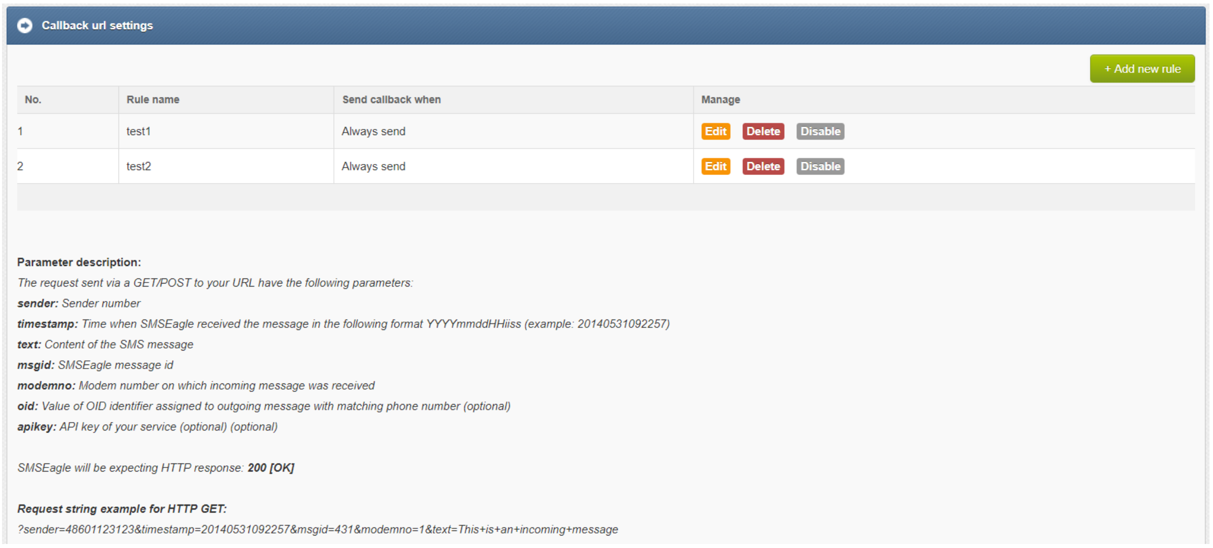
- if you want to use the plugin, set ‘Enable forwarding to email’ to ‘Yes’
- choose a type of email forwarding: “To email of last sending user” (so called “Two-way Email2SMS & SMS&Email”) or “To fixed email address”
- enter an email address to which incoming SMS messages are to sent
- enter SMTP configuration for your SMTP server that will be used for sending emails

Callback URL plugin

Callback URL plugin allows you to forward incoming SMS message to a defined URL address. If the plugin is enabled, on each incoming SMS message SMSEagle will trigger HTTP(S) request to a defined URL. HTTP(S) request can be of type GET or POST.

PLUGIN CONFIGURATION

Plugin "Callback URL" allows to add unlimited number of rules. Each rule can be enabled or disabled by user.



No.	Rule name	Send callback when	Manage
1	test1	Always send	Edit Delete Disable
2	test2	Always send	Edit Delete Disable

Parameter description:
The request sent via a GET/POST to your URL have the following parameters:
sender: *Sender number*
timestamp: *Time when SMSEagle received the message in the following format YYYYmmddHHiiss (example: 20140531092257)*
text: *Content of the SMS message*
msgid: *SMSEagle message id*
modemno: *Modem number on which incoming message was received*
oid: *Value of OID identifier assigned to outgoing message with matching phone number (optional)*
apikey: *API key of your service (optional) (optional)*

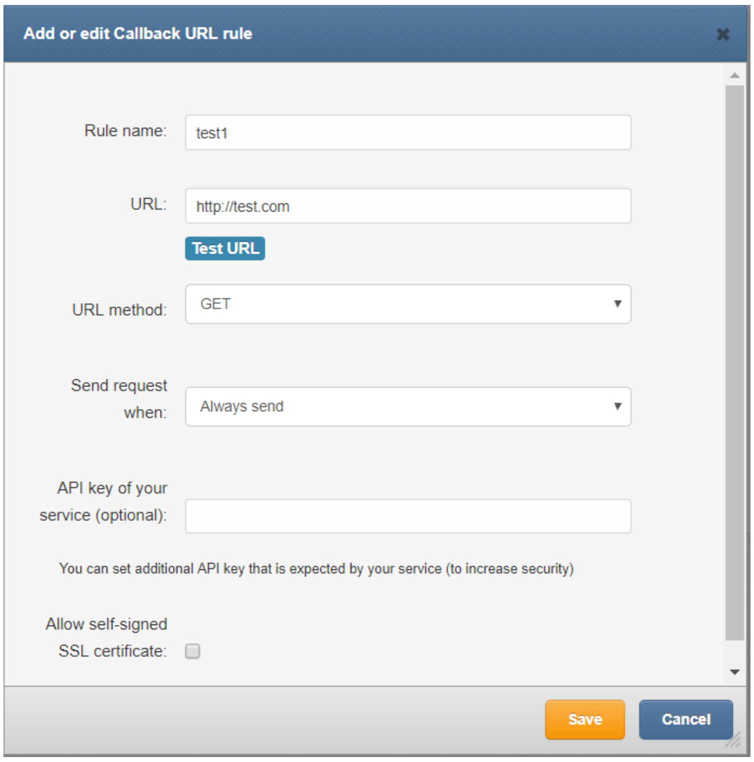
SMSEagle will be expecting HTTP response: ***200 [OK]***

Request string example for HTTP GET:
```
?sender=48601123123&timestamp=20140531092257&msgid=431&modemno=1&text=This+is+an+incoming+message
```

Screenshot from Callback URL settings

For each new rule user has to fill in the requested fields:

- 'URL' field defines remote address of your callback script
- 'Test URL' button allows to test whether your Callback URL configuration is correct. SMSEagle will make a callback request with test parameters and will verify the response of remote server
- 'URL method' allows to choose whether callback to your URL is done with HTTP(S) GET or POST method
- "Send request when" defines if the request is always sent, sent only when SMS sender belongs to a given contact/group or only when incoming message contains a given character string
- Optionally you can define "API key of your service" value. This will be passed to your callback URL in parameter 'apikey'. If you leave the field blank, 'apikey' parameter will not be passed to your callback URL
- User may also choose whether to enable support of self-signed SSL certificate



Rule name:	test1
URL:	http://test.com
URL method:	GET
Send request when:	Always send
API key of your service (optional):	
Allow self-signed SSL certificate:	☐

You can set additional API key that is expected by your service (to increase security)

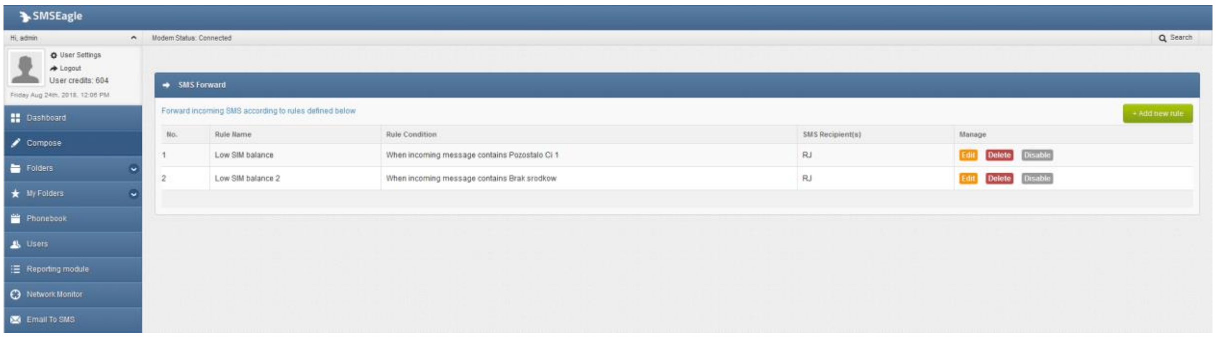
After sending HTTP(S) GET/POST request to your callback URL, SMSEagle will be expecting HTTP response: 200 [OK]. If other or no response is received from your callback URL, SMSEagle will keep retrying every 2 minutes for 24 hours.

SMS Forward

The plugin "SMS forward" allows to forward incoming SMS messages to one/may recipients according to defined rules.

PLUGIN CONFIGURATION

Plugin "SMS Forward" allows to add many forwarding rules. Each rule can be enabled or disabled by user.



Forward incoming SMS according to rules defined below

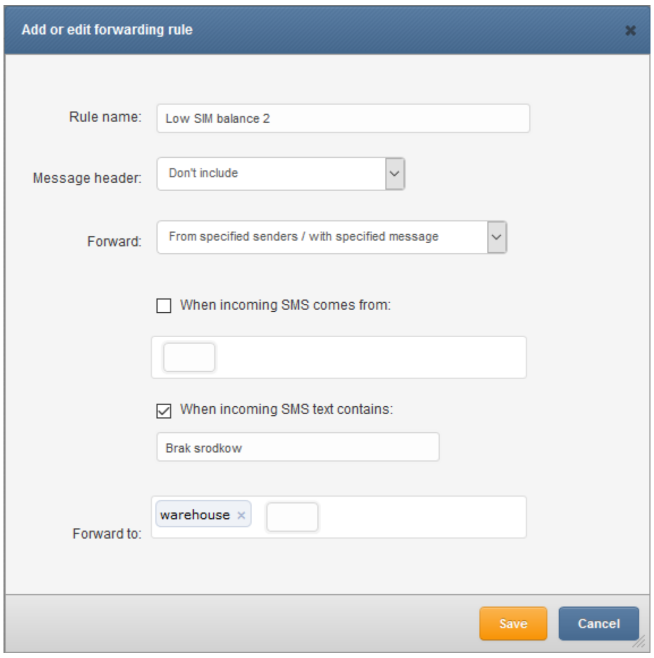
No.	Rule Name	Rule Condition	SMS Recipient(s)	Manage
1	Low SIM balance	When incoming message contains Pozostalo Ci 1	RJ	Edit Delete Disable
2	Low SIM balance 2	When incoming message contains Brak srodkow	RJ	Edit Delete Disable

Screenshot from plugin main window

For each rule user can define:

- When incoming SMS should be forwarded (Rule type) and to what number(s) the message should be forwarded (SMS Recipient).
- Whether or not include in SMS a sender number from which original SMS came from.

- When defining a rule user can choose SMS recipient (who gets the forwarded SMS).
- It can be either phone number or name of group from phonebook.
- User may define many forwarding rules in the plugin.
- Each rule is processed independently.
- There is a possibility to enable/disable each rule.


Add or edit forwarding rule

Rule name:	Low SIM balance 2
Message header:	Don't include
Forward:	From specified senders / with specified message
	☐ When incoming SMS comes from:
	☑ When incoming SMS text contains:
	Brak srodkow
Forward to:	warehouse ×

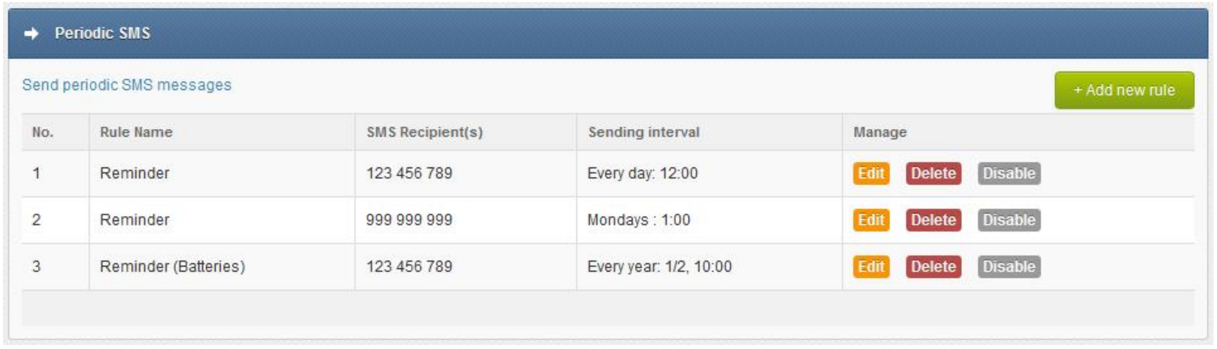
Screenshot form "Add/edit forwarding rule"

Periodic SMS

The plugin "Periodic SMS" allows to send SMS messages or USSD codes at a desired time interval. User may define many sending rules, and each rule will be processed independently.

PLUGIN CONFIGURATION

Plugin "Periodic SMS" allows to add many sending rules. Each rule can be enabled or disabled by user.


Periodic SMS

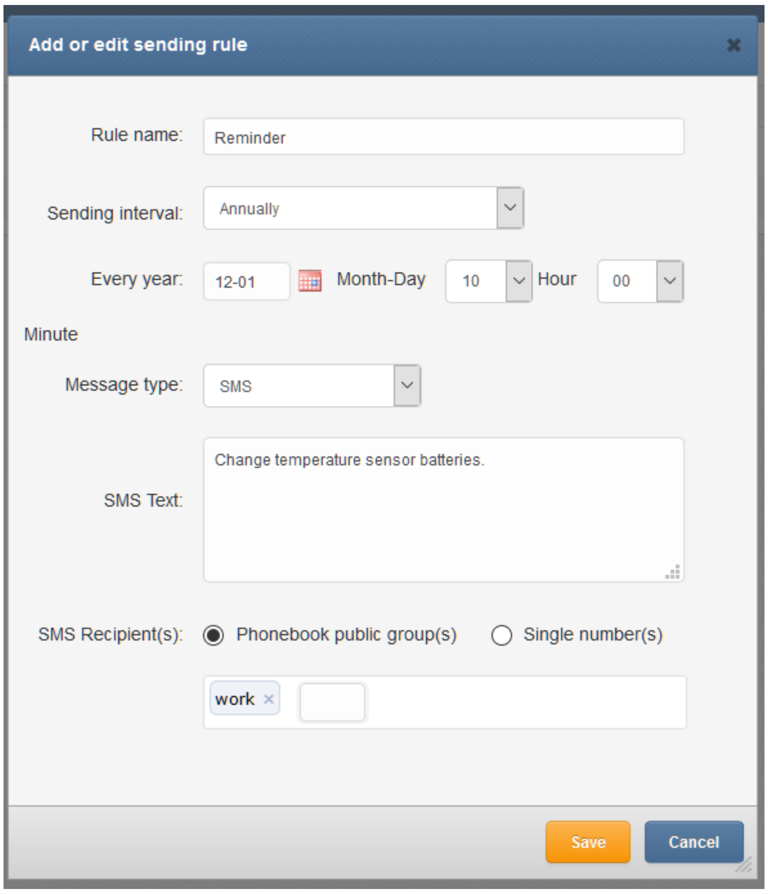
Send periodic SMS messages

No.	Rule Name	SMS Recipient(s)	Sending interval	Manage
1	Reminder	123 456 789	Every day: 12:00	Edit Delete Disable
2	Reminder	999 999 999	Mondays : 1:00	Edit Delete Disable
3	Reminder (Batteries)	123 456 789	Every year: 1/2, 10:00	Edit Delete Disable

Screenshot from plugin main window

For each rule the user can define:

- The rule name
- Sending interval (Hourly, Daily, Weekly, Monthly or Annually)
- Message type (SMS, USSD Code)
- The content of the SMS text
- The recipients, sending SMS to a single number or a group from phonebook



Rule name: Reminder

Sending interval: Annually

Every year: 12-01 Month-Day 10 Hour 00 Minute

Message type: SMS

SMS Text: Change temperature sensor batteries.

SMS Recipient(s): [✓ Phonebook public group(s)] Single number(s)

work ×

Screenshot from "Add new rule" window

Digital input/output

The NPE-9300 device is equipped with 4 usable digital inputs (DI) and 2 digital outputs (DO). The digital inputs can be used to receive signals from outside sensors or devices and automatically trigger sending of SMS message based on input state. On the other hand the digital outputs may be used to activate external devices connected to the outputs when certain SMS messages are received by SMSEagle.

The digital inputs of SMSEagle NPE-9300 devices are of type pull-up (*see more https://en.wikipedia.org/wiki/Pull-up_resistor*) and are represented by the following states:

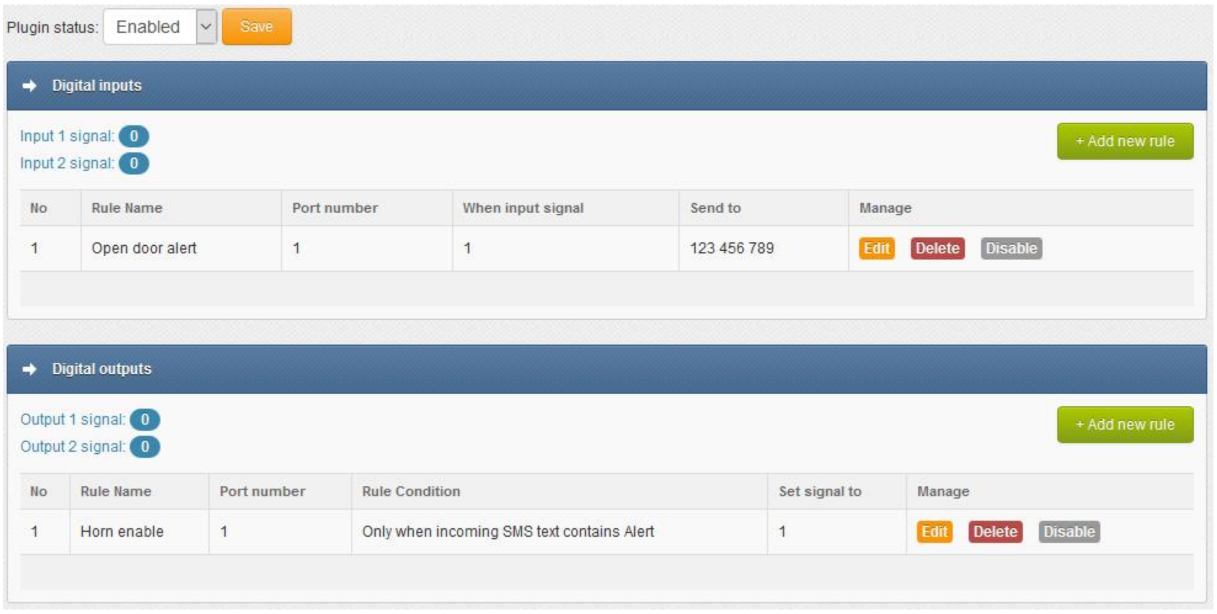
Logical level DI	Voltage
LOW (0)	GROUND
HIGH (1)	OPEN CIRCUIT

The digital outputs of SMSEagle NPE-9300 devices are of type open-collector (*see more https://en.wikipedia.org/wiki/Open_collector*) and are represented by the following states:

Logical level DO	Voltage
LOW (0)	GROUND
HIGH (1)	OPEN CIRCUIT

PLUGIN CONFIGURATION

The plugin “Digital input/output” allows you to define rules that control the behaviour of digital inputs/outputs on SMSEagle device. User may define several processing rules for both inputs and outputs.



No	Rule Name	Port number	When input signal	Send to	Manage
1	Open door alert	1	1	123 456 789	Edit Delete Disable

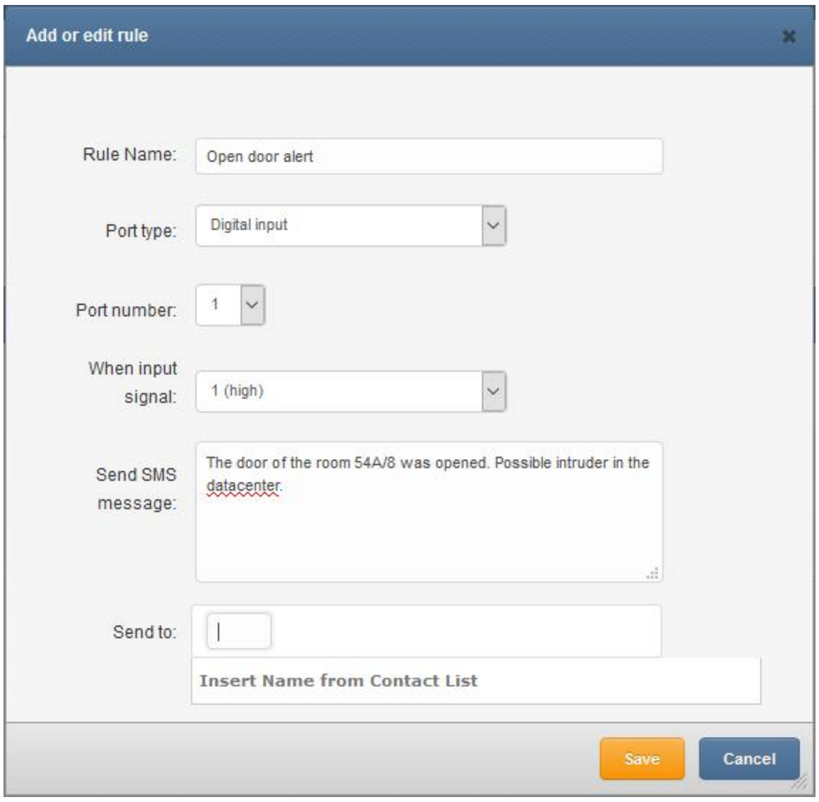
No	Rule Name	Port number	Rule Condition	Set signal to	Manage
1	Horn enable	1	Only when incoming SMS text contains Alert	1	Edit Delete Disable

Screenshot from plugin window

DIGITAL INPUTS

For each processing rule for digital input user can define:

- The rule name
- Port number (1…7)
- State of input signal that will trigger sending of SMS message (field “When input signal”)
- SMS text (field “Send SMS message”)
- The recipients from phonebook (field “Send to”)



Rule Name: Open door alert

Port type: Digital input

Port number: 1

When input signal: 1 (high)

Send SMS message: The door of the room 54A/8 was opened. Possible intruder in the datacenter.

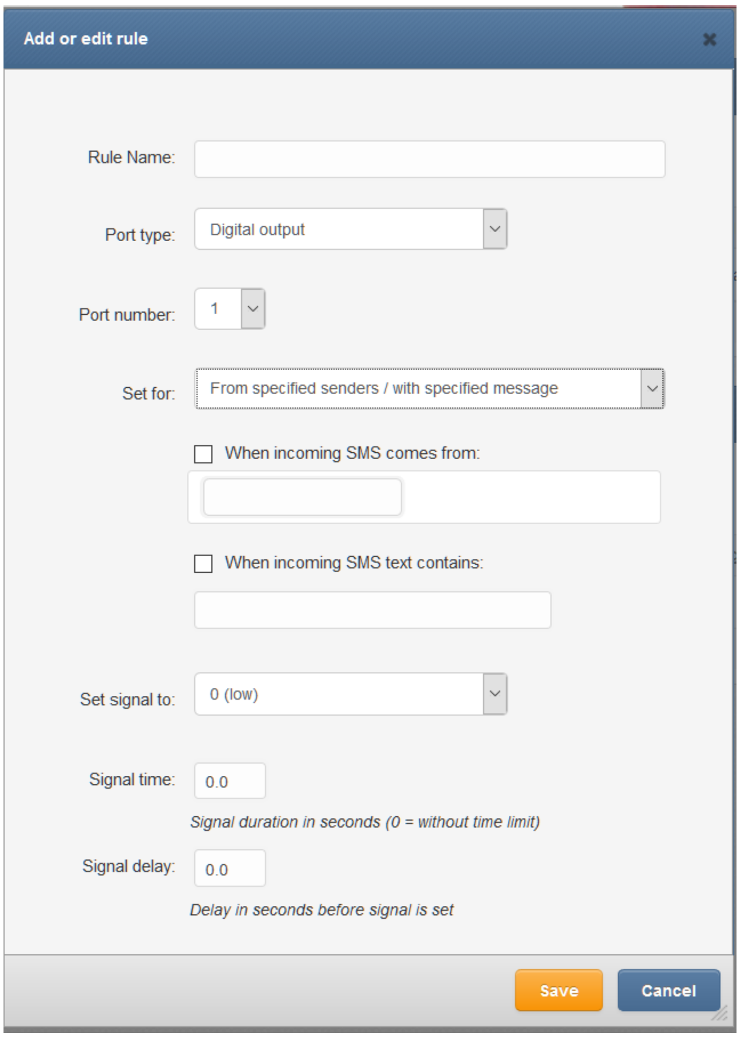
Send to:

Screenshot from digital input "Add or edit rule" window

DIGITAL OUTPUTS

For each processing rule for digital output user can define:

- The rule name
- Port number (1,2)
- On what condition digital output should be set (all incoming messages, when incoming SMS comes from specified contact in phonebook or when incoming SMS text contains given value)
- State of output signal that will be triggered by incoming SMS message
- Output signal duration in seconds (0 = without time limit)
- Output signal delay before signal is set



Add or edit rule

Rule Name:	
Port type:	Digital output
Port number:	1
Set for:	From specified senders / with specified message
	☐ When incoming SMS comes from:
	☐ When incoming SMS text contains:
Set signal to:	0 (low)
Signal time:	0.0
	Signal duration in seconds (0 = without time limit)
Signal delay:	0.0
	Delay in seconds before signal is set

Screenshot from digital output "Add or edit rule" window

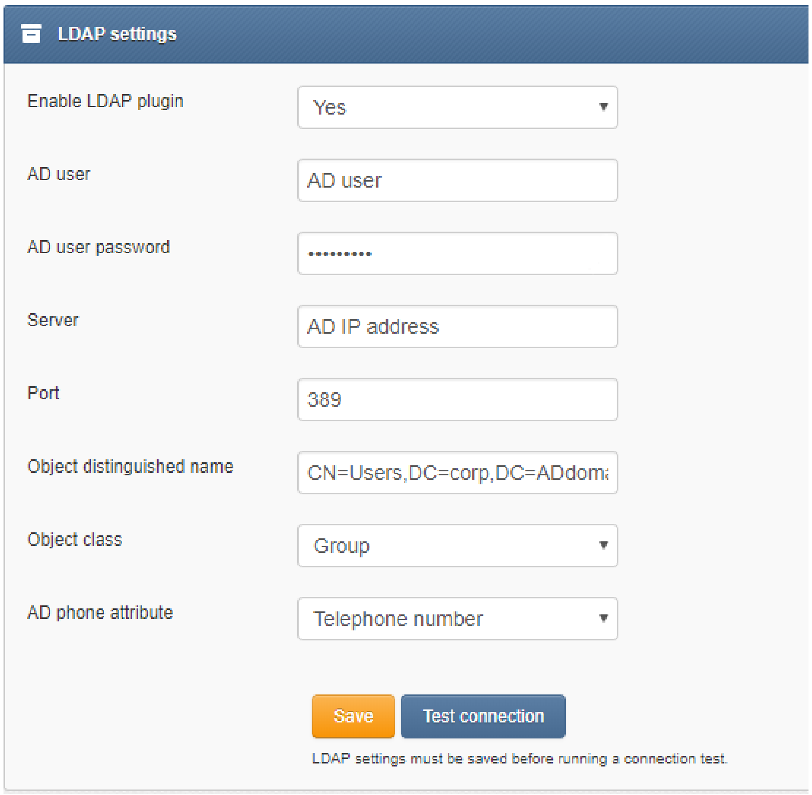
LDAP plugin

The LDAP plugin allows to access Active Directory (hereinafter referred to as "AD") and read contacts and groups in SMSEagle web-GUI.

PLUGIN CONFIGURATION

Choose "LDAP" from left side menu in SMSEagle web-GUI to access plugin configuration. After enabling the plugin, user needs to fill in all requested fields according to AD settings.

In the "AD phone attribute" field user needs to choose which phone attribute from AD will be shown in SMSEagle web-gui.

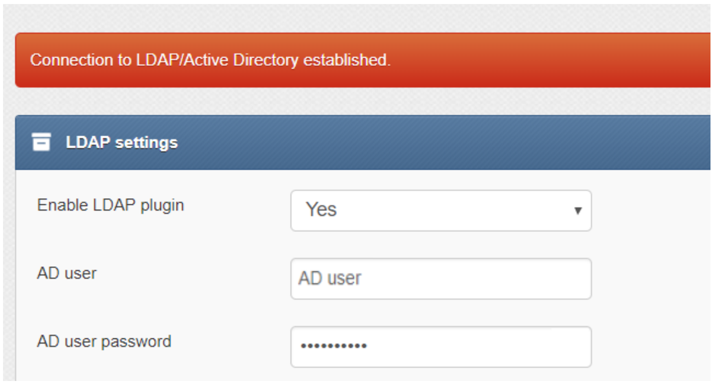

LDAP settings

Enable LDAP plugin	Yes
AD user	AD user
AD user password	•••••••••
Server	AD IP address
Port	389
Object distinguished name	CN=Users,DC=corp,DC=ADdom
Object class	Group
AD phone attribute	Telephone number

LDAP settings must be saved before running a connection test.

Screenshot from "LDAP settings" window

Click "Save" and "Test connection" to make sure that SMSEagle is connected with AD server.


Connection to LDAP/Active Directory established.

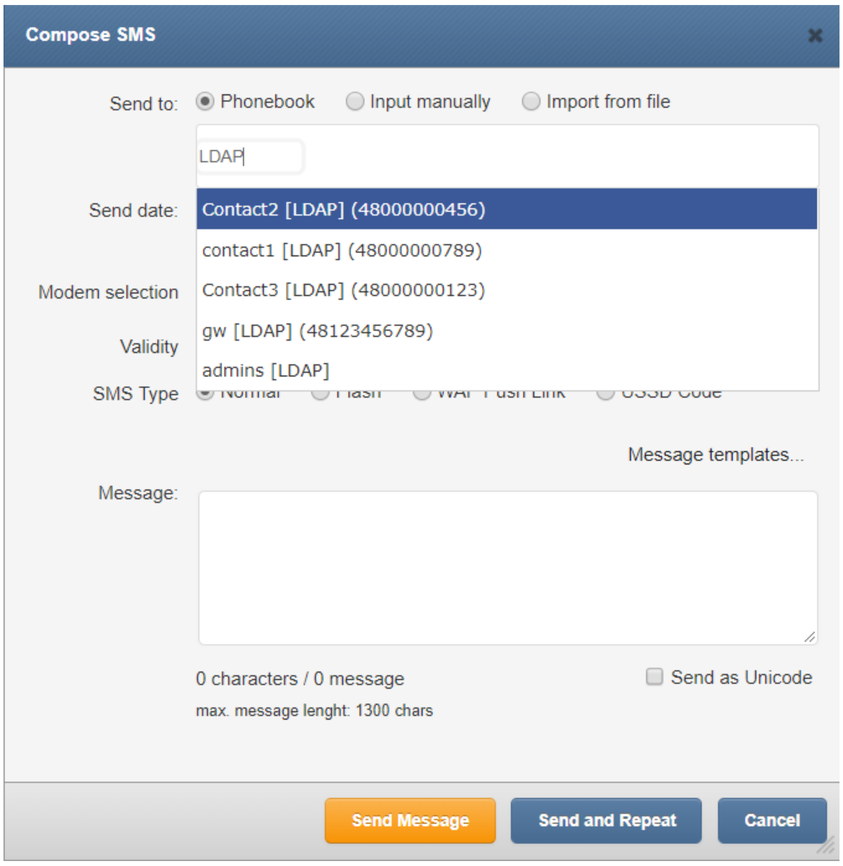
LDAP settings

Enable LDAP plugin	Yes
AD user	AD user
AD user password	••••••••••

Screenshot showing successful connection to AD server.

With connection established, AD contacts/groups suggestions are shown in selected modules of web-gui. Start typing any part of contact/group name or number to show AD contact suggestions.

Type “LDAP” (case sensitive) to check all contacts listed in AD directory.


Compose SMS

Send to: [✓ Phonebook] Input manually Import from file

LDAP

Contact2 [LDAP] (48000000456)
contact1 [LDAP] (48000000789)
Contact3 [LDAP] (48000000123)
gw [LDAP] (48123456789)
admins [LDAP]

Send date:

Modem selection

Validity

SMS Type

Message templates...

Message:

0 characters / 0 message

max. message lenght: 1300 chars

Send as Unicode

Screenshot from “Compose” module with LDAP connection enabled

LDAP directory suggestions can be used in “Compose”, “Autoreply” and “Digital input/output” modules.

SMSEagle API

SMSEagle has powerful built-in HTTP API functionalities. REST API is dedicated for integration of SMSEagle with any external system or application. Below you will find a detailed description of API functionalities.

Please note, that SMSEagle API supports both HTTP and HTTPS protocol.

For your convenience sample usage of SMSEagle's API in most popular programming languages are available at: http://www.smseagle.eu/code-samples/

1. Send SMS: HTTP GET method

HTTP GET METHOD:

```
https://url-of-smseagle/index.php/http_api/send_sms
```

PARAMETERS:

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
to	recipient telephone number (or numbers separated with comma)
message	your SMS message
date	*(optional parameter)* date and time in format YYYYmmDDHHMM (YYYY – year, mm – month, DD – day, HH – hour, MM – minute). If this parameter is not null SMS will be scheduled for sending at the given date and time
highpriority	*(optional parameter)* 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	*(optional parameter)* 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	*(optional parameter)* 0 = normal SMS (default), 1 = SMS will be sent as flash message
oid	*(optional parameter)* This attribute specifies a user-defined unique ID that is assigned to a message-recipient pair. The oid is a varchar(36) that uniquely identifies a message sent to a particular recipient (particular phone number). The value of this ID allows client applications to match incoming reply messages to outgoing messages. If no oid was assigned to the outgoing message this attribute will have a value of null for incoming message. The oid value will be automatically assigned to incoming message only if incoming phone number matches exactly the phone number (including country code) from outgoing message.
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/send_sms?
login=john&pass=doe&to=1234567&message=mymessage

https://url-of-smseagle/index.php/http_api/send_sms?
login=john&pass=doe&to=1234567&message=mymessage&date=201401152132

https://url-of-smseagle/index.php/http_api/send_sms?
login=john&pass=doe&to=1234567&message=mymessage&highpriority=1
```

RESPONSE:

Response: **OK; ID=[ID of message in outbox]**
Sample response: OK; ID=297

Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):

Response:
```
<xml>
   <message_id>[ID of message in outbox]</message_id>
   <status>ok</status>
</xml>
```

Sample response:
```
<xml>
   <message_id>297</message_id>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong parameters</error_text>
   <status>error</status>
</xml>
```

Important notice: You must encode URL before sending it to gateway if you use national characters in SMS message text.

2. Send SMS: JSONRPC method

HTTP POST METHOD:
```
https://url-of-smseagle/index.php/jsonrpc/sms
```

PARAMETERS:

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
to	recipient telephone number (or numbers separated with comma)
message	your SMS message
date	*(optional parameter)* date and time in format YYYYmmDDHHMM (YYYY – year, mm – month, DD – day, HH – hour, MM – minute). If this parameter is not null SMS will be scheduled for sending at the given date and time
highpriority	*(optional parameter)* 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	*(optional parameter)* 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	(optional parameter) 0 = normal SMS (default), 1 = SMS will be sent as flash message
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

SAMPLE BODY:
```
{"method":"sms.send_sms",
"params":{"login":"john","pass":"doe","to":"481234567","message":"My
message"}}
or
{"method":"sms.send_sms",
"params":{"login":"john","pass":"doe","to":"481234567","message":"My
message","date":"201401152132"}}
or
{"method":"sms.send_sms",
"params":{"login":"john","pass":"doe","to":"481234567","message":"My
message","highpriority":"1"}}
```

RESPONSE:

Response: `{"result": "OK; ID=[ID of message in outbox]"}`
Sample response: `{"result": "OK; ID=297"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when wrong parameters): `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):

Response:
`{"result": {"message_id":"[ID of message in outbox]","status":"ok"}}`

Sample response: `{"result": {"message_id":"748","status":"ok"}}`

Response (when wrong logindata):

```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong parameters","status":"error"}}
```

3. Send SMS to a group: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/send_togroup
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
groupname	group name defined in your SMSEagle Phonebook. The group must be defined as Public
message	your SMS message
date	*(optional parameter)* date and time in format YYYYmmDDHHMM (YYYY – year, mm – month, DD – day, HH – hour, MM – minute). If this parameter is not null SMS will be scheduled for sending at the given date and time
highpriority	*(optional parameter)* 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	*(optional parameter)* 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	*(optional parameter)* 0 = normal SMS (default), 1 = SMS will be sent as flash message
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/send_togroup?
login=john&pass=doe&groupname=admins&message=mymessage
```

```
https://url-of-smseagle/index.php/http_api/send_togroup?
login=john&pass=doe&groupname=admins&message=mymessage&date=20140115213
2
```

```
https://url-of-smseagle/index.php/http_api/send_togroup?
login=john&pass=doe&groupname=admins&message=mymessage&highpriority=1
```

RESPONSE:

Response: **OK; ID=[ID of message in outbox]**

Sample response: OK; ID=297

Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):

Response:
```
<xml>
   <message_id>[ID of message in outbox]</message_id>
   <status>ok</status>
</xml>
```

Sample response:
```
<xml>
   <message_id>297</message_id>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong parameters</error_text>
   <status>error</status>
</xml>
```

4. Send SMS to a group: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
groupname	group name defined in your SMSEagle Phonebook. The group must be defined as Public
message	your SMS message
date	*(optional parameter)* date and time in format YYYYmmDDHHMM (YYYY – year, mm – month, DD – day, HH – hour, MM – minute). If this parameter is not null SMS will be scheduled for sending at the given date and time
highpriority	*(optional parameter)* 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	*(optional parameter)* 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	*(optional parameter)* 0 = normal SMS (default), 1 = SMS will be sent as flash message

modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:
```
{"method":"sms.send_togroup",
"params":{"login":"john","pass":"doe","groupname":"admins","message":"m
ymessage"}}
or
{"method":"sms.send_togroup",
"params":{"login":"john","pass":"doe","groupname":"admins","message":"m
ymessage","date":"201401152132"}}
or
{"method":"sms.send_togroup",
"params":{"login":"john","pass":"doe","groupname":"admins","message":"m
ymessage","highpriority":"1"}}
```

RESPONSE:

Response: `{"result": "OK; ID=[ID of message in outbox]"}`

Sample response: `{"result": "OK; ID=[297]"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters): `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):

Response:
`{"result": {"message_id":"[ID of message in outbox]","status":"ok"}}`

Sample response: `{"result": {"message_id":"748","status":"ok"}}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`

Response (when wrong parameters):
`{"result": {"error_text":"Wrong parameters","status":"error"}}`

5. Send SMS to contact: HTTP GET method

HTTP GET METHOD:
`https://url-of-smseagle/index.php/http_api/send_tocontact`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
contactname	contact name (or names separated by comma) defined in your SMSEagle Phonebook . Contacts must be defined as Public

message	your SMS message
date	*(optional parameter)* date and time in format YYYYmmDDHHMM (YYYY – year, mm – month, DD – day, HH – hour, MM – minute). If this parameter is not null SMS will be scheduled for sending at the given date and time
highpriority	*(optional parameter)* 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	*(optional parameter)* 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	*(optional parameter)* 0 = normal SMS (default), 1 = SMS will be sent as flash message
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/send_tocontact?
login=john&pass=doe&contactname=johndoe&message=mymessage
```

```
https://url-of-smseagle/index.php/http_api/send_tocontact?
login=john&pass=doe&contactname=johndoe&message=mymessage&date=20140115
2132
```

```
https://url-of-smseagle/index.php/http_api/send_tocontact?
login=john&pass=doe&contactname=johndoe&message=mymessage&highpriority=
1
```

RESPONSE:

Response: **OK; ID=[ID of message in outbox]**

Sample response: OK; ID=297

Response (when wrong logindata): **Invalid login or password**

Response (when contact doesn't exist): **Invalid contact name – [contact_name]**

Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):

Response:

```
<xml>
  <item>
    <message_id>[ID of message in outbox]</message_id>
    <status>ok</status>
  </item>
</xml>
```

Sample response:

```
<xml>
  <item>
    <message_id>297</message_id>
```

```
      <status>ok</status>
   </item>
</xml>

Response (when wrong logindata):
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>

Response (when contact doesn’t exist):
<xml>
   <item>
      <error_text>Invalid contact name – [contact_name]</error_text>
      <status>error</status>
   </item>
</xml>


Response (when wrong parameters):
<xml>
   <error_text>Wrong parameters</error_text>
   <status>error</status>
</xml>
```

6. Send SMS to contact: JSONRPC method

HTTP POST METHOD:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
contactname	contact name defined in your SMSEagle Phonebook. The contact must be defined as Public
message	your SMS message
date	*(optional parameter)* date and time in format YYYYmmDDHHMM (YYYY – year, mm – month, DD – day, HH – hour, MM – minute). If this parameter is not null SMS will be scheduled for sending at the given date and time
highpriority	*(optional parameter)* 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	*(optional parameter)* 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	*(optional parameter)* 0 = normal SMS (default), 1 = SMS will be sent as flash message
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"sms.send_tocontact",
"params":{"login":"john","pass":"doe","contactname":"johndoe","message"
:"mymessage"}}
or
{"method":"sms.send_tocontact",
"params":{"login":"john","pass":"doe","contactname":"johndoe","message"
:"mymessage","date":"201401152132"}}
or
{"method":"sms.send_tocontact",
"params":{"login":"john","pass":"doe","contactname":"johndoe","message"
:"mymessage","highpriority":"1"}}
```

RESPONSE:

Response: `{"result": "OK; ID=[ID of message in outbox]"}`
Sample response: `{"result": "OK; ID=[297]"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when contact doesn't exist): `{"result": "Invalid contact name – contact_name]"}`
Response (when wrong parameters): `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):

Response:
`{"result": [{"message_id":"[ID of message in outbox]","status":"ok"}]}`

Sample response: `{"result": [{"message_id":"748","status":"ok"}]}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`

Response (when contact doesn't exist):
```
{"result": [{"error_text":"Invalid contact name –
contact_name]","status":"error"}]}
```

Response (when wrong parameters):
`{"result": {"error_text":"Wrong parameters","status":"error"}}`

7. Send USSD code: HTTP GET method

HTTP GET METHOD:
`https://url-of-smseagle/index.php/http_api/send_ussd`

PARAMETERS:

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle

to	USSD code (url-encoded)
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/send_ussd?
login=john&pass=doe&to= %2A101%23
```

RESPONSE:

Response: **OK; ID=[ID of message in outbox]**
Sample response: OK; ID=297

Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):

Response:

```
<xml>
   <message_id>[ID of message in outbox]</message_id>
   <status>ok</status>
</xml>
```

Sample response:

```
<xml>
   <message_id>297</message_id>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
   <error_text>Wrong parameters</error_text>
   <status>error</status>
</xml>
```

Important notice: You must urlencode USSD code before sending it to gateway. Result will show up on device Inbox folder.

8. Send USSD code: JSONRPC method

HTTP POST METHOD:
```
https://url-of-smseagle/index.php/jsonrpc/sms
```

PARAMETERS:

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
to	USSD code
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

SAMPLE BODY:
```
{"method":"sms.send_ussd",
"params":{"login":"john","pass":"doe","to":"*101#"}}
```

RESPONSE:
Response: `{"result": "OK; ID=[ID of message in outbox]"}`
Sample response: `{"result": "OK; ID=297"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when wrong parameters): `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):
Response:
```
{"result": {"message_id":"[ID of message in outbox]","status":"ok"}}
```

Sample response: `{"result": {"message_id":"748","status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong parameters","status":"error"}}
```

Important notice: Result will show up on device Inbox folder.

9. Send binary SMS: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/send_binary_sms
```

PARAMETERS:

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle

to	recipient telephone number (or numbers separated with comma)
udh	*(optional parameter)* UDH header for the message (in hex format)
data	binary message (in hex format)
class	*(optional parameter)* message class
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/send_binary_sms?
login=john&pass=doe&to=1234567&udh=0605040B8423F0&data=EA0601AE02056A00
45C60C037777772E736D736561676C652E657500080103534D534561676C65000101
```

RESPONSE:

Response: **OK; ID=[ID of message in outbox]**

Sample response: OK; ID=297

Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong or missing >>udh<< parameter**
Response (when wrong parameters): **Wrong or missing >>data<< parameter**

RESPONSE (XML):

Response:
<xml>
  <message_id>**[ID of message in outbox]**</message_id>
  <status>ok</status>
</xml>

Sample response:
<xml>
  <message_id>297</message_id>
  <status>ok</status>
</xml>

Response (when wrong logindata):
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>

Response (when wrong parameters):
<xml>
  <error_text> Wrong or missing >>udh<< parameter </error_text>
  <status>error</status>
</xml>

Response (when wrong parameters):
<xml>
  <error_text> Wrong or missing >>data<< parameter </error_text>
  <status>error</status>

</xml>

10. Send binary SMS: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

PARAMETERS:

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
to	recipient telephone number (or numbers separated with comma)
udh	*(optional parameter)* UDH header for the message (in hex format)
data	binary message (in hex format)
class	*(optional parameter)* message class
modem_no	*(optional parameter)* sending modem number (only for multimodem devices)
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"sms.send_binary_sms",
"params":{"login":"john","pass":"doe","to":"1234567","udh":"0605040B842
3F0","data":"EA0601AE02056A0045C60C037777772E736D736561676C652E65750008
0103534D534561676C65000101"}}
```

RESPONSE:

Response: `{"result": "OK; ID=[ID of message in outbox]"}`
Sample response: `{"result": "OK; ID=297"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when wrong parameters): `{"result": "Wrong or missing >>udh<< parameter"}`
Response (when wrong parameters): `{"result": "Wrong or missing >>data<< parameter"}`

RESPONSE (EXTENDED):

Response:
`{"result": {"message_id":"[ID of message in outbox]","status":"ok"}}`

Sample response: `{"result": {"message_id":"748","status":"ok"}}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`

Response (when wrong parameters):
```
{"result": {"error_text":" Wrong or missing >>udh<< parameter
","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>data<< parameter",
"status":"error"}}
```

11. Read SMS: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/read_sms
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
folder	one of the following: inbox, outbox, sentitems
idfrom	*(optional parameter)* minimal message-id
idto	*(optional parameter)* maximum message-id
from	*(optional parameter)* telephone number of SMS sender (for inbox)
to	*(optional parameter)* telephone number of SMS receiver (for sentitems)
datefrom	*(optional parameter)* date and time in format YYYYmmDDHHMMSS (YYYY – year, mm – month, DD – day, HH – hour, MM – minutes, SS – seconds). Show only messages sent/received on this date/time and later
dateto	*(optional parameter)* date and time in format YYYYmmDDHHMMSS (YYYY – year, mm – month, DD – day, HH – hour, MM – minutes, SS – seconds). Show only messages sent/received on this date/time and earlier
limit	*(optional parameter)* how many messages to show
unread	*(optional parameter)* 1 = show only unread messages
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
Show all messages from inbox:
https://url-of-smseagle/index.php/http_api/read_sms?
login=john&pass=doe&folder=inbox

Show all unread messages from inbox:
https://url-of-smseagle/index.php/http_api/read_sms?
login=john&pass=doe&folder=inbox&unread=1

Show messages from sentitems folder with id=1234 to 1236:
https://url-of-smseagle/index.php/http_api/read_sms?
login=john&pass=doe&folder=sentitems&idfrom=1234&idto=1236
```

```
Show messages from inbox folder with sender phone number +481234567:
https://url-of-smseagle/index.php/http_api/read_sms?
login=john&pass=doe&folder=inbox&from=+481234567

Show messages from sentitems folder with receiver phone number 7654321
and datetime from 2014-12-24 08:10:00 to 2014-12-31 23:59:59:
https://url-of-smseagle/index.php/http_api/read_sms?
login=john&pass=doe&folder=sentitems&to=7654321&datefrom=20141224081000
&dateto=20141231235959
```

RESPONSE:

Sample responses: inbox folder, sentitems folder

Response (when no data): **No data to display**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):

Sample response (inbox folder):

```
<xml>
    <messages>
       <item>
            <UpdatedInDB>2018-07-17 15:11:31</UpdatedInDB>
            <ReceivingDateTime>2018-07-17 15:04:04</ReceivingDateTime>
            <Text>00540065007300740020031</Text>
            <SenderNumber>+48123456789</SenderNumber>
            <Coding>Default_No_Compression</Coding>
            <UDH></UDH>
            <SMSCNumber>+48790998250</SMSCNumber>
            <Class>-1</Class>
            <TextDecoded>Test 1</TextDecoded>
            <ID>124</ID>
            <RecipientID>smseagle1</RecipientID>
            <Processed>t</Processed>
            <id_folder>1</id_folder>
            <readed>true</readed>
            <oid></oid>
            <Status>0</Status>
        </item>
        <item>
            <UpdatedInDB>2018-07-17 15:11:31</UpdatedInDB>
            <ReceivingDateTime>2018-07-17 15:04:10</ReceivingDateTime>
            <Text>00540065007300740020032</Text>
            <SenderNumber>+48123456788</SenderNumber>
            <Coding>Default_No_Compression</Coding>
            <UDH></UDH>
            <SMSCNumber>+48790998250</SMSCNumber>
            <Class>-1</Class>
            <TextDecoded>Test 2</TextDecoded>
            <ID>125</ID>
            <RecipientID>smseagle1</RecipientID>
            <Processed>t</Processed>
            <id_folder>1</id_folder>
            <readed>true</readed>
            <oid>5208facc-5912-4d21-8d31-7f830cf8f24e</oid>
            <Status>0</Status>
        </item>
        <item>
            <UpdatedInDB>2018-07-17 15:11:31</UpdatedInDB>
            <ReceivingDateTime>2018-07-17 15:05:49</ReceivingDateTime>

<Text>004C006F00720065006D0020006900700073075006D00200064006F006C006F0
07200200073006900740020061006D00650074002C00200063006F006E007300650063
0074006500740075007200200061006400690070006900730063069006E00670020006
5006C00690074002E0020004300720061007300200066006500720006D0065006E007400
75006D00200075006C006C0061006D0063006F00720070006500720020006500670650
07300740061007300 2E0020004E0075006C006C006100200070006C006100630065007 2
0061007400200066 0069006E006900620075007300200064006F006C006F0072002C002
0006D0061006C00650073007500610064006100200076006100720069007500730020000
6C00690067007500 6C0061002000680065006E006400720065</Text>
            <SenderNumber>+48123456787</SenderNumber>
            <Coding>Default_No_Compression</Coding>
            <UDH>050003590301</UDH>
            <SMSCNumber>+48790998250</SMSCNumber>
            <Class>-1</Class>
            <TextDecoded>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Cras fermentum ullamcorper egestas. Nulla placerat
finibus dolor, malesuada varius ligula hendrerit sed. Nullam nisl
sapien, molestie rhoncus orci vel, viverra luctus ipsum. Praesent
maximus luctus orci. Vestibulum lacus dui, vestibulum ac aliquam eget,
ultrices et mi. In ac felis urna. Phasellus eget leo a leo congue
ultricies. Donec tincidunt volutpat arcu a commodo</TextDecoded>
            <ID>126</ID>
            <RecipientID>smseagle1</RecipientID>
            <Processed>t</Processed>
            <id_folder>1</id_folder>
            <readed>true</readed>
            <oid></oid>
            <Status>0</Status>
        </item>
      </messages>
    <status>ok</status>
</xml>
```

Sample response (sentitems folder):

```
<xml>
    <messages>
        <item>
            <UpdatedInDB>2018-06-07 11:29:56</UpdatedInDB>
            <InsertIntoDB>2018-06-07 11:29:43</InsertIntoDB>
            <SendingDateTime>2018-06-07 11:29:56</SendingDateTime>
            <DeliveryDateTime>2018-06-07 11:30:05</DeliveryDateTime>
            <Text>0074006500730074</Text>
            <DestinationNumber>123456789</DestinationNumber>
            <Coding>Default_No_Compression</Coding>
            <UDH></UDH>
            <SMSCNumber>+48501200777</SMSCNumber>
            <Class>-1</Class>
            <TextDecoded>test</TextDecoded>
            <ID>456</ID>
            <SenderID>smseagle1</SenderID>
            <SequencePosition>1</SequencePosition>
        <Status>DeliveryOK</Status>
        <StatusError>-1</StatusError>
        <TPMR>116</TPMR>
        <RelativeValidity>255</RelativeValidity>
        <CreatorID>admin</CreatorID>
        <id_folder>3</id_folder>
        <StatusCode>-1</StatusCode>
    </item>
    <item>
        <UpdatedInDB>2018-07-13 11:40:45</UpdatedInDB>
        <InsertIntoDB>2018-07-13 11:40:40</InsertIntoDB>
        <SendingDateTime>2018-07-13 11:40:45</SendingDateTime>
        <DeliveryDateTime></DeliveryDateTime>
        <Text></Text>
        <DestinationNumber>*101#</DestinationNumber>
        <Coding>8bit</Coding>
        <UDH></UDH>
        <SMSCNumber>+48501200777</SMSCNumber>
        <Class>127</Class>
        <TextDecoded></TextDecoded>
        <ID>525</ID>
        <SenderID>smseagle1</SenderID>
        <SequencePosition>1</SequencePosition>
        <Status>SendingOK</Status>
        <StatusError>-1</StatusError>
        <TPMR>-1</TPMR>
        <RelativeValidity>255</RelativeValidity>
        <CreatorID>admin</CreatorID>
        <id_folder>3</id_folder>
        <StatusCode>-1</StatusCode>
    </item>
    <item>
        <UpdatedInDB>2018-07-18 14:25:41</UpdatedInDB>
        <InsertIntoDB>2018-07-18 14:25:23</InsertIntoDB>
        <SendingDateTime>2018-07-18 14:25:28</SendingDateTime>
        <DeliveryDateTime>2018-07-18 14:25:28</DeliveryDateTime>
        <Text>00540065007300740020007400650073007400031</Text>
        <DestinationNumber>+48123456788</DestinationNumber>
        <Coding>Default_No_Compression</Coding>
        <UDH></UDH>
        <SMSCNumber>+48601000310</SMSCNumber>
        <Class>-1</Class>
        <TextDecoded>Test test1</TextDecoded>
        <ID>574</ID>
        <SenderID>smseagle1</SenderID>
        <SequencePosition>1</SequencePosition>
        <Status>DeliveryOK</Status>
        <StatusError>0</StatusError>
        <TPMR>84</TPMR>
        <RelativeValidity>255</RelativeValidity>
        <CreatorID>admin</CreatorID>
        <id_folder>3</id_folder>
        <StatusCode>-1</StatusCode>
    </item>
    <item>
        <UpdatedInDB>2018-07-18 14:27:13</UpdatedInDB>
        <InsertIntoDB>2018-07-18 14:27:03</InsertIntoDB>
        <SendingDateTime>2018-07-18 14:27:13</SendingDateTime>
        <DeliveryDateTime></DeliveryDateTime>
```

```
<Text>00540065007300740020007700690074006800200075006E00690063006F00640
06500200065006E0063006F00640069006E0067003A00200105014200F30119017A0107
</Text>
            <DestinationNumber>123456788</DestinationNumber>
            <Coding>Unicode_No_Compression</Coding>
            <UDH></UDH>
            <SMSCNumber>+48601000310</SMSCNumber>
            <Class>-1</Class>
            <TextDecoded>Test with unicode encoding:
ąłóęźć</TextDecoded>
            <ID>576</ID>
            <SenderID>smseagle2</SenderID>
            <SequencePosition>1</SequencePosition>
            <Status>SendingOK</Status>
            <StatusError>-1</StatusError>
            <TPMR>86</TPMR>
            <RelativeValidity>255</RelativeValidity>
            <CreatorID>admin</CreatorID>
            <id_folder>3</id_folder>
            <StatusCode>-1</StatusCode>
        </item>
        <item>
            <UpdatedInDB>2018-07-18 14:27:36</UpdatedInDB>
            <InsertIntoDB>2018-07-18 14:27:32</InsertIntoDB>
            <SendingDateTime>2018-07-18 14:27:36</SendingDateTime>
            <DeliveryDateTime></DeliveryDateTime>

<Text>00540065007300740020006F006600200066006C0061007300680020006D00650
0730073006100670065</Text>
            <DestinationNumber>123456788</DestinationNumber>
            <Coding>Default_No_Compression</Coding>
            <UDH></UDH>
            <SMSCNumber>+48601000310</SMSCNumber>
            <Class>0</Class>
            <TextDecoded>Test of flash message</TextDecoded>
            <ID>577</ID>
            <SenderID>smseagle2</SenderID>
            <SequencePosition>1</SequencePosition>
            <Status>SendingOK</Status>
            <StatusError>-1</StatusError>
            <TPMR>87</TPMR>
            <RelativeValidity>255</RelativeValidity>
            <CreatorID>admin</CreatorID>
            <id_folder>3</id_folder>
            <StatusCode>-1</StatusCode>
        </item>
        <item>
            <UpdatedInDB>2018-07-18 14:29:29</UpdatedInDB>
            <InsertIntoDB>2018-07-18 14:28:46</InsertIntoDB>
            <SendingDateTime>2018-07-18 14:29:29</SendingDateTime>
            <DeliveryDateTime></DeliveryDateTime>

<Text>00540065007300740020007700690074006800200062006100640020007000680
06F006E00650020006E0075006D006200650072</Text>
            <DestinationNumber>11</DestinationNumber>
            <Coding>Default_No_Compression</Coding>
            <UDH></UDH>
            <SMSCNumber></SMSCNumber>
            <Class>-1</Class>
            <TextDecoded>Test with bad phone number</TextDecoded>
            <ID>578</ID>
            <SenderID>smseagle2</SenderID>
            <SequencePosition>1</SequencePosition>
            <Status>SendingError</Status>
            <StatusError>-1</StatusError>
            <TPMR>-1</TPMR>
            <RelativeValidity>255</RelativeValidity>
            <CreatorID>admin</CreatorID>
            <id_folder>3</id_folder>
            <StatusCode>21</StatusCode>
        </item>
    </messages>
    <status>ok</status>
</xml>
```

Response (when no data):

```
<xml>
  <error_text>No data to display</error_text>
  <status>error</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
  <error_text>Wrong parameters</error_text>
  <status>error</status>
</xml>
```

FIELD DESCRIPTION OF RESPONSE DATA – INBOX FOLDER:

Field	Data type	Description
UpdatedInDB	timestamp	when somebody (software, user) updated the message content or state
ReceivingDateTime	timestamp	when SMS was received
Text	text	SMS text encoded using hex values
SenderNumber	character varying(30)	SMS sender number
Coding	character varying(255)	SMS text coding. Possible values: *'Default_No_Compression',* *'Unicode_No_Compression',* *'8bit',* *'Default_Compression',* *'Unicode_Compression'*
UDH	text	User Data Header encoded using hex values
SMSCNumber	character varying(20)	SMSC number
Class	integer	SMS class (0 is flash SMS, -1 is normal SMS, 127 is USSD)
TextDecoded	text	decoded SMS text
ID	serial	SMS unique identification number

RecipientID	text	which modem received the message *(for example: smseagle1, smseagle2)*
Processed	boolean	whether SMS was processed by SMSEagle application
id_folder	integer	identification of storage folder. Possible values: *1 Inbox* *5 Trash* *11… Custom folder*
readed	text	whether SMS was read in GUI or via API
oid	character varying(36)	user-defined unique ID that is assigned to a message-recipient pair. The oid uniquely identifies a message sent to a particular recipient (particular phone number). *More information: see send_sms method description*
Status	integer	Status of incoming message. Currently only used for USSD messages with following meaning: *1 Unknown status.* *2 No action is needed, maybe network initiated USSD.* *3 Reply is expected.* *4 USSD dialog terminated.* *5 Another client replied.* *6 Operation not supported.* *7 Network timeout.*

FIELD DESCRIPTION OF RESPONSE DATA – SENTITEMS FOLDER:

Field	Data type	Description
UpdatedInDB	timestamp	when somebody (software, user) updated the message content or state
InsertIntoDB	timestamp	when message was inserted into database
SendingDateTime	timestamp	when message has been sent
DeliveryDateTime	timestamp	time of receiving a delivery report (if it has been enabled). Null if delivery report was not received.
Text	text	SMS text encoded using hex values
DestinationNumber	character varying(30)	destination number for SMS
Coding	character varying(255)	SMS text coding. Possible values: *'Default_No_Compression',* *'Unicode_No_Compression',* *'8bit',* *'Default_Compression',* *'Unicode_Compression'*
UDH	text	User Data Header encoded using hex values
SMSCNumber	character varying(20)	number of SMSC, which sent SMS
Class	integer	SMS class (0 is flash SMS, -1 is normal SMS, 127 is USSD)
TextDecoded	text	decoded SMS text
ID	serial	SMS unique identification number
SenderID	character varying(255)	which modem sent the message *(for example: smseagle1, smseagle2)*
SequencePosition	integer	SMS number in SMS sequence
Status	character varying(255)	Status of message sending. Possible values: *SendingOK* *Message has been sent, waiting for delivery report*

		SendingOKNoReport *Message has been sent without asking for delivery report* *SendingError* *Sending has failed* *DeliveryOK* *Delivery report arrived and reported success* *DeliveryFailed* *Delivery report arrived and reports failure* *DeliveryPending* *Delivery report announced pending deliver* *DeliveryUnknown* *Delivery report reported unknown status* *Error* *Some other error happened during sending* *Notice: some cellular operators return "SendingOK" status instead of "DeliveryOK" for correctly delivered SMS. If you want to check for delivery status, please verify what you receive from your operator or instead use the field DeliveryDateTime.*
StatusError	integer	Status of delivery from delivery report message, codes are defined in GSM specification 03.40 section 9.2.3.15 (TP-Status)
TPMR	integer	The Message Reference field (TP-MR) as defined in GSM 03.40
RelativeValidity	integer	SMS relative validity (TP-VP) encoded as defined in GSM 03.40
CreatorID	text	username that created the SMS message
id_folder	integer	identification of storage folder. Possible values: *3 Sent items* *5 Trash* *11… Custom folder*
StatusCode	integer	CMS status code (also known as CMS ERROR) received from cellular network. - *1 No CMS Error* > *-1 CMS Error occurred. CMS error number is saved in this field.*

12. Read SMS: JSONRPC method

HTTP POST METHOD:

```
https://url-of-smseagle/index.php/jsonrpc/sms
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
folder	one of the following: inbox, outbox, sentitems
idfrom	*(optional parameter)* minimal message-id
idto	*(optional parameter)* maximum message-id
from	*(optional parameter)* telephone number of SMS sender (for inbox)

to	*(optional parameter)* telephone number of SMS receiver (for sentitems)
datefrom	*(optional parameter)* date and time in format YYYYmmDDHHMMSS (YYYY – year, mm – month, DD – day, HH – hour, MM – minutes, SS – seconds). Show only messages sent/received on this date/time and later
dateto	*(optional parameter)* date and time in format YYYYmmDDHHMMSS (YYYY – year, mm – month, DD – day, HH – hour, MM – minutes, SS – seconds). Show only messages sent/received on this date/time and earlier
limit	*(optional parameter)* how many messages to show
unread	*(optional parameter)* 1 = show only unread messages
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
Show all messages from inbox:
{"method":"sms.read_sms",
"params":{"login":"john","pass":"doe","folder":"inbox"}}

Show all unread messages from inbox:
{"method":"sms.read_sms",
"params":{"login":"john","pass":"doe","folder":"inbox","unread":"1"}}

Show messages from sentitems folder with id=1234 to 1236:
{"method":"sms.read_sms",
"params":{"login":"john","pass":"doe","folder":"sentitems","idfrom":"12
34","idto":"1236"}}

Show messages from inbox folder with sender phone number +481234567:
{"method":"sms.read_sms",
"params":{"login":"john","pass":"doe","folder":"inbox","from":"
481234567"}}

Show messages from sentitems folder with receiver phone number 7654321
and datetime from 2014-12-24 08:10:00 to 2014-12-31 23:59:59:
{"method":"sms.read_sms",
"params":{"login":"john","pass":"doe","folder":"sentitems","to":"765432
1","datefrom":"20141224081000","dateto":"20141231235959"}}
```

RESPONSE:

Sample response (inbox folder):

```
{
    "result": [
        {
            "UpdatedInDB": "2018-07-18 13:56:16",
            "ReceivingDateTime": "2018-07-17 15:04:04",
            "Text": "00540065007300740020031",
            "SenderNumber": "+48123456789",
            "Coding": "Default_No_Compression",
            "UDH": "",
            "SMSCNumber": "+48790998250",
            "Class": "-1",
            "TextDecoded": "Test 1",
            "ID": "124",
            "RecipientID": "smseagle1",
            "Processed": "t",
            "id_folder": "1",
            "readed": "true",
            "oid": "",
            "Status": "0"
        },
        {
            "UpdatedInDB": "2018-07-18 13:56:16",
            "ReceivingDateTime": "2018-07-17 15:04:10",
            "Text": "00540065007300740020032",
            "SenderNumber": "+48123456788",
            "Coding": "Default_No_Compression",
            "UDH": "",
            "SMSCNumber": "+48790998250",
            "Class": "-1",
            "TextDecoded": "Test 2",
            "ID": "125",
            "RecipientID": "smseagle1",
            "Processed": "t",
            "id_folder": "1",
            "readed": "true",
            "oid": "5208facc-5912-4d21-8d31-7f830cf8f24e",
            "Status": "0"
        },
        {
            "UpdatedInDB": "2018-07-18 13:56:16",
            "ReceivingDateTime": "2018-07-17 15:05:49",
            "Text":
"004C006F00720065006D0020006900700073007500 6D00200064006F006C006F007200
20007300690074002000061006D00650074002C00200063006F006E0073006500630074 0
06500740075007200200061006400690070006900730063006900 6E0067002000650 06C
00690074002E002000430072006100730020006600650072006D0065006E00740075006
D00200075006C006C0061006D0063006F0072007000650072002000650067006500730 0
7400610073002E0020004E0075006C006C00610020007000 6C00610063006500720061 0
074002000660069006E0069006200750073002000640 06F006C006F0072002C0020006D
0061006C0065007300750061006400610020007600610072006900750073002000 6C006
900670075006C0061002000680065006E006400720065",
            "SenderNumber": "+48123456787",
            "Coding": "Default_No_Compression",
            "UDH": "050003590301",
            "SMSCNumber": "+48790998250",
            "Class": "-1",
            "TextDecoded": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Cras fermentum ullamcorper egestas. Nulla placerat
finibus dolor, malesuada varius ligula hendrerit sed. Nullam nisl
sapien, molestie rhoncus orci vel, viverra luctus ipsum. Praesent
maximus luctus orci. Vestibulum lacus dui, vestibulum ac aliquam eget,
ultrices et mi. In ac felis urna. Phasellus eget leo a leo congue
ultricies. Donec tincidunt volutpat arcu a commodo",
            "ID": "126",
            "RecipientID": "smseagle1",
            "Processed": "t",
            "id_folder": "1",
            "readed": "true",
            "oid": "",
            "Status": "0"
        }
    ]
}
```

Sample response (sentitems folder):

```
{
    "result": [
        {
            "UpdatedInDB": "2018-06-07 11:29:56",
            "InsertIntoDB": "2018-06-07 11:29:43",
            "SendingDateTime": "2018-06-07 11:29:56",
            "DeliveryDateTime": "2018-06-07 11:30:05",
            "Text": "0074006500730074",
            "DestinationNumber": "+48123456789",
            "Coding": "Default_No_Compression",
            "UDH": "",
            "SMSCNumber": "+48501200777",
            "Class": "-1",
            "TextDecoded": "test",
            "ID": "456",
            "SenderID": "smseagle1",
            "SequencePosition": "1",
            "Status": "DeliveryOK",
            "StatusError": "-1",
            "TPMR": "116",
            "RelativeValidity": "255",
            "CreatorID": "admin",
            "id_folder": "3",
            "StatusCode": "-1"
        },
        {
            "UpdatedInDB": "2018-07-13 11:40:45",
            "InsertIntoDB": "2018-07-13 11:40:40",
            "SendingDateTime": "2018-07-13 11:40:45",
            "DeliveryDateTime": null,
            "Text": "",
            "DestinationNumber": "*101#",
            "Coding": "8bit",
            "UDH": "",
            "SMSCNumber": "+48501200777",
            "Class": "127",
            "TextDecoded": "",
            "ID": "525",
            "SenderID": "smseagle1",
            "SequencePosition": "1",
            "Status": "SendingOK",
            "StatusError": "-1",
            "TPMR": "-1",
            "RelativeValidity": "255",
            "CreatorID": "admin",
            "id_folder": "3",
            "StatusCode": "-1"
        },
        {
            "UpdatedInDB": "2018-07-18 14:25:41",
            "InsertIntoDB": "2018-07-18 14:25:23",
            "SendingDateTime": "2018-07-18 14:25:28",
            "DeliveryDateTime": "2018-07-18 14:25:28",
            "Text": "00540065007300740020007400650073007400031",
            "DestinationNumber": "+48123456788",
            "Coding": "Default_No_Compression",
            "UDH": "",
            "SMSCNumber": "+48601000310",
            "Class": "-1",
            "TextDecoded": "Test test1",
            "ID": "574",
            "SenderID": "smseagle1",
            "SequencePosition": "1",
            "Status": "DeliveryOK",
            "StatusError": "0",
            "TPMR": "84",
            "RelativeValidity": "255",
            "CreatorID": "admin",
            "id_folder": "3",
            "StatusCode": "-1"
        },
        {
            "UpdatedInDB": "2018-07-18 14:27:13",
            "InsertIntoDB": "2018-07-18 14:27:03",
            "SendingDateTime": "2018-07-18 14:27:13",
            "DeliveryDateTime": null,
            "Text":
"00540065007300740020007700690074006800200075006E00690063006F0064006500
200065006E0063006F00640069006E0067003A00200105014200F30119017A0107",
            "DestinationNumber": "123456788",
            "Coding": "Unicode_No_Compression",
            "UDH": "",
            "SMSCNumber": "+48601000310",
            "Class": "-1",
            "TextDecoded": "Test with unicode encoding: ąłóęźć",
            "ID": "576",
            "SenderID": "smseagle1",
            "SequencePosition": "1",
            "Status": "SendingOK",
            "StatusError": "-1",
            "TPMR": "86",
            "RelativeValidity": "255",
            "CreatorID": "admin",
            "id_folder": "3",
            "StatusCode": "-1"
        },
        {
            "UpdatedInDB": "2018-07-18 14:27:36",
            "InsertIntoDB": "2018-07-18 14:27:32",
            "SendingDateTime": "2018-07-18 14:27:36",
            "DeliveryDateTime": null,
            "Text":
"005400650073007400200066F006600200066006C006100730068002000 6D0065007300
73006100670065",
            "DestinationNumber": "123456788",
            "Coding": "Default_No_Compression",
            "UDH": "",
            "SMSCNumber": "+48601000310",
            "Class": "0",
            "TextDecoded": "Test of flash message",
            "ID": "577",
            "SenderID": "smseagle1",
            "SequencePosition": "1",
            "Status": "SendingOK",
            "StatusError": "-1",
            "TPMR": "87",
            "RelativeValidity": "255",
            "CreatorID": "admin",
            "id_folder": "3",
            "StatusCode": "-1"
        },
        {
            "UpdatedInDB": "2018-07-18 14:29:29",
            "InsertIntoDB": "2018-07-18 14:28:46",
            "SendingDateTime": "2018-07-18 14:29:29",
            "DeliveryDateTime": null,
            "Text":
"00540065007300740020007700690074006800200062006100640020007000680 06F00
6E00650020006E0075006D006200650072",
            "DestinationNumber": "11",
            "Coding": "Default_No_Compression",
            "UDH": "",
            "SMSCNumber": "",
            "Class": "-1",
            "TextDecoded": "Test with bad phone number",
            "ID": "578",
            "SenderID": "smseagle1",
            "SequencePosition": "1",
            "Status": "SendingError",
            "StatusError": "-1",
            "TPMR": "-1",
            "RelativeValidity": "255",
            "CreatorID": "admin",
            "id_folder": "3",
            "StatusCode": "21"
        }
    ]
}
```

Response (when no data): `{"result": "No data to display"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters): `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):

Sample response (inbox folder):

```
{
    "result": {
        "messages": [
            {
                "UpdatedInDB": "2018-07-18 14:06:06",
                "ReceivingDateTime": "2018-07-17 15:04:04",
                "Text": "0054006500730074002000031",
                "SenderNumber": "+48123456789",
                "Coding": "Default_No_Compression",
                "UDH": "",
                "SMSCNumber": "+48790998250",
                "Class": "-1",
                "TextDecoded": "Test 1",
                "ID": "124",
                "RecipientID": "smseagle1",
                "Processed": "t",
                "id_folder": "1",
                "readed": "true",
                "oid": "",
                "Status": "0"
            },
            {
                "UpdatedInDB": "2018-07-18 14:06:06",
                "ReceivingDateTime": "2018-07-17 15:04:10",
                "Text": "005400650073007400200032",
                "SenderNumber": "+48123456788",
                "Coding": "Default_No_Compression",
                "UDH": "",
                "SMSCNumber": "+48790998250",
                "Class": "-1",
                "TextDecoded": "Test 2",
                "ID": "125",
                "RecipientID": "smseagle1",
                "Processed": "t",
                "id_folder": "1",
                "readed": "true",
                "oid": "5208facc-5912-4d21-8d31-7f830cf8f24e",
                "Status": "0"
            },
            {
                "UpdatedInDB": "2018-07-18 14:06:06",
                "ReceivingDateTime": "2018-07-17 15:05:49",
                "Text":
"004C006F00720065006D00200069007000730075006D00200064006F006C006F007200
2000730069007400200061006D00650074002C00200063006F006E00730065006300740
065007400750072002000610064006900700069007300630069006E006700200065006C
00690074002E002000430072006100730020006600650072006D0065006E00740075006
D00200075006C006C0061006D0063006F00720070006500720020006500670065007300
7400610073002E0020004E0075006C006C0061002000700006C006100630065007200610
074002000660069006E006900620075007300200064006F006C006F0072002C0020006D
0061006C00650073007500610064006100200076006100720069007500730020006C006
900670075006C0061002000680065006E006400720065",
                "SenderNumber": "+48123456787",
                "Coding": "Default_No_Compression",
                "UDH": "050003590301",
                "SMSCNumber": "+48790998250",
                "Class": "-1",
                "TextDecoded": "Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Cras fermentum ullamcorper egestas. Nulla placerat
finibus dolor, malesuada varius ligula hendrerit sed. Nullam nisl
sapien, molestie rhoncus orci vel, viverra luctus ipsum. Praesent
maximus luctus orci. Vestibulum lacus dui, vestibulum ac aliquam eget,
ultrices et mi. In ac felis urna. Phasellus eget leo a leo congue
ultricies. Donec tincidunt volutpat arcu a commodo",
                "ID": "126",
                "RecipientID": "smseagle1",
                "Processed": "t",
                "id_folder": "1",
                "readed": "true",
                "oid": "",
                "Status": "0"
            }
        ],
        "status": "ok"
```

```
    }
}
```

Sample response (sentitems folder):

```
{
    "result": {
        "messages": [
            {
                "UpdatedInDB": "2018-06-07 11:29:56",
                "InsertIntoDB": "2018-06-07 11:29:43",
                "SendingDateTime": "2018-06-07 11:29:56",
                "DeliveryDateTime": "2018-06-07 11:30:05",
                "Text": "0074006500730074",
                "DestinationNumber": "+48123456789",
                "Coding": "Default_No_Compression",
                "UDH": "",
                "SMSCNumber": "+48501200777",
                "Class": "-1",
                "TextDecoded": "test",
                "ID": "456",
                "SenderID": "smseagle1",
                "SequencePosition": "1",
                "Status": "DeliveryOK",
                "StatusError": "-1",
                "TPMR": "116",
                "RelativeValidity": "255",
                "CreatorID": "admin",
                "id_folder": "3",
                "StatusCode": "-1"
            },
            {
                "UpdatedInDB": "2018-07-13 11:40:45",
                "InsertIntoDB": "2018-07-13 11:40:40",
                "SendingDateTime": "2018-07-13 11:40:45",
                "DeliveryDateTime": null,
                "Text": "",
                "DestinationNumber": "*101#",
                "Coding": "8bit",
                "UDH": "",
                "SMSCNumber": "+48501200777",
                "Class": "127",
                "TextDecoded": "",
                "ID": "525",
                "SenderID": "smseagle1",
                "SequencePosition": "1",
                "Status": "SendingOK",
                "StatusError": "-1",
                "TPMR": "-1",
                "RelativeValidity": "255",
                "CreatorID": "admin",
                "id_folder": "3",
                "StatusCode": "-1"
            },
            {
                "UpdatedInDB": "2018-07-18 14:25:41",
                "InsertIntoDB": "2018-07-18 14:25:23",
                "SendingDateTime": "2018-07-18 14:25:28",
                "DeliveryDateTime": "2018-07-18 14:25:28",
                "Text": "00540065007300740020007400650073007400031",
                "DestinationNumber": "+48123456788",
                "Coding": "Default_No_Compression",
                "UDH": "",
                "SMSCNumber": "+48601000310",
                "Class": "-1",
                "TextDecoded": "Test test1",
                "ID": "574",
                "SenderID": "smseagle1",
                "SequencePosition": "1",
                "Status": "DeliveryOK",
                "StatusError": "0",
                "TPMR": "84",
                "RelativeValidity": "255",
                "CreatorID": "admin",
                "id_folder": "3",
                "StatusCode": "-1"
            },
            {
                "UpdatedInDB": "2018-07-18 14:27:13",
                "InsertIntoDB": "2018-07-18 14:27:03",
                "SendingDateTime": "2018-07-18 14:27:13",
                "DeliveryDateTime": null,
                "Text":
"0054006500730074002000770069007400680020007500 6E00690063006F0064006500
200065006E0063006F00640069006E0067003A00200105014200F30119017A0107",
                "DestinationNumber": "123456788",
                "Coding": "Unicode_No_Compression",
                "UDH": "",
                "SMSCNumber": "+48601000310",
                "Class": "-1",
                "TextDecoded": "Test with unicode encoding: ąłóęźć",
                "ID": "576",
                "SenderID": "smseagle1",
                "SequencePosition": "1",
                "Status": "SendingOK",
                "StatusError": "-1",
                "TPMR": "86",
                "RelativeValidity": "255",
                "CreatorID": "admin",
                "id_folder": "3",
                "StatusCode": "-1"
            },
            {
                "UpdatedInDB": "2018-07-18 14:27:36",
                "InsertIntoDB": "2018-07-18 14:27:32",
                "SendingDateTime": "2018-07-18 14:27:36",
                "DeliveryDateTime": null,
                "Text":
"00540065007300740020006F00660020006600 6C0061007300680020006D0065007300
73006100670065",
                "DestinationNumber": "123456788",
                "Coding": "Default_No_Compression",
                "UDH": "",
                "SMSCNumber": "+48601000310",
                "Class": "0",
                "TextDecoded": "Test of flash message",
                "ID": "577",
                "SenderID": "smseagle1",
                "SequencePosition": "1",
                "Status": "SendingOK",
                "StatusError": "-1",
                "TPMR": "87",
                "RelativeValidity": "255",
                "CreatorID": "admin",
                "id_folder": "3",
                "StatusCode": "-1"
            },
            {
                "UpdatedInDB": "2018-07-18 14:29:29",
                "InsertIntoDB": "2018-07-18 14:28:46",
                "SendingDateTime": "2018-07-18 14:29:29",
                "DeliveryDateTime": null,
                "Text":
"0054006500730074002000770069007400680020006200610064002000700068006F00
6E00650020006E0075006D006200650072",
                "DestinationNumber": "11",
                "Coding": "Default_No_Compression",
                "UDH": "",
                "SMSCNumber": "",
                "Class": "-1",
                "TextDecoded": "Test with bad phone number",
                "ID": "578",
                "SenderID": "smseagle1",
                "SequencePosition": "1",
                "Status": "SendingError",
                "StatusError": "-1",
                "TPMR": "-1",
                "RelativeValidity": "255",
                "CreatorID": "admin",
                "id_folder": "3",
                "StatusCode": "21"
            }
        ],
        "status": "ok"
}
```

Response (when no data):
```
{"result": {"error_text":" No data to display ", "status":"error"}}
```

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":" Wrong or missing >>udh<< parameter
","status":"error"}}
```

FIELD DESCRIPTION OF RESPONSE DATA – INBOX FOLDER:

Field	Data type	Description
UpdatedInDB	timestamp	when somebody (software, user) updated the message content or state
ReceivingDateTime	timestamp	when SMS was received
Text	text	SMS text encoded using hex values
SenderNumber	character varying(30)	SMS sender number
Coding	character varying(255)	SMS text coding. Possible values:

		'Default_No_Compression', *'Unicode_No_Compression',* *'8bit',* *'Default_Compression',* *'Unicode_Compression'*
UDH	text	User Data Header encoded using hex values
SMSCNumber	character varying(20)	SMSC number
Class	integer	SMS class (0 is flash SMS, -1 is normal SMS, 127 is USSD)
TextDecoded	text	decoded SMS text
ID	serial	SMS unique identification number
RecipientID	text	which modem received the message *(for example: smseagle1, smseagle2)*
Processed	boolean	whether SMS was processed by SMSEagle application
id_folder	integer	identification of storage folder. Possible values: *1 Inbox* *5 Trash* *11… Custom folder*
readed	text	whether SMS was read in GUI or via API
oid	character varying(36)	user-defined unique ID that is assigned to a message-recipient pair. The oid uniquely identifies a message sent to a particular recipient (particular phone number). *More information: see send_sms method description*
Status	integer	Status of incoming message. Currently only used for USSD messages with following meaning: *1 Unknown status.* *2 No action is needed, maybe network initiated USSD.* *3 Reply is expected.* *4 USSD dialog terminated.* *5 Another client replied.* *6 Operation not supported.* *7 Network timeout.*

FIELD DESCRIPTION OF RESPONSE DATA – SENTITEMS FOLDER:

Field	Data type	Description
UpdatedInDB	timestamp	when somebody (software, user) updated the message content or state
InsertIntoDB	timestamp	when message was inserted into database
SendingDateTime	timestamp	when message has been sent
DeliveryDateTime	timestamp	time of receiving a delivery report (if it has been enabled). Null if delivery report was not received.
Text	text	SMS text encoded using hex values
DestinationNumber	character varying(30)	destination number for SMS
Coding	character varying(255)	SMS text coding. Possible values: *'Default_No_Compression',* *'Unicode_No_Compression',* *'8bit',* *'Default_Compression',* *'Unicode_Compression'*
UDH	text	User Data Header encoded using hex values

SMSCNumber	character varying(20)	number of SMSC, which sent SMS
Class	integer	SMS class (0 is flash SMS, -1 is normal SMS, 127 is USSD)
TextDecoded	text	decoded SMS text
ID	serial	SMS unique identification number
SenderID	character varying(255)	which modem sent the message *(for example: smseagle1, smseagle2)*
SequencePosition	integer	SMS number in SMS sequence
Status	character varying(255)	Status of message sending. Possible values: *SendingOK* *Message has been sent, waiting for delivery report* *SendingOKNoReport* *Message has been sent without asking for delivery report* *SendingError* *Sending has failed* *DeliveryOK* *Delivery report arrived and reported success* *DeliveryFailed* *Delivery report arrived and reports failure* *DeliveryPending* *Delivery report announced pending deliver* *DeliveryUnknown* *Delivery report reported unknown status* *Error* *Some other error happened during sending* *Notice: some cellular operators return "SendingOK" status instead of "DeliveryOK" for correctly delivered SMS. If you want to check for delivery status, please verify what values you receive from your operator or instead use the field DeliveryDateTime.*
StatusError	integer	Status of delivery from delivery report message, codes are defined in GSM specification 03.40 section 9.2.3.15 (TP-Status)
TPMR	integer	The Message Reference field (TP-MR) as defined in GSM 03.40
RelativeValidity	integer	SMS relative validity (TP-VP) encoded as defined in GSM 03.40
CreatorID	text	username that created the SMS message
id_folder	integer	identification of storage folder. Possible values: *3 Sent items* *5 Trash* *11... Custom folder*
StatusCode	integer	CMS status code (also known as CMS ERROR) received from cellular network. *- 1 No CMS Error* *> -1 CMS Error occurred. CMS error number is saved in this field.*

13. Delete SMS: HTTP GET method

HTTP GET METHOD:

```
https://url-of-smseagle/index.php/http_api/delete_sms
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
folder	one of the following: inbox, outbox, sentitems
idfrom	minimal id of message
idto	maximal id of message
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
Delete message with id=1234 from inbox:
https://url-of-smseagle/index.php/http_api/delete_sms?
login=john&pass=doe&folder=inbox&idfrom=1234&idto=1234

Delete messages with id 1234 - 1250 from inbox:
https://url-of-smseagle/index.php/http_api/delete_sms?
login=john&pass=doe&folder=inbox&idfrom=1234&idto=1250

Delete all messages from outbox:
https://url-of-smseagle/index.php/http_api/delete_sms?
login=john&pass=doe&folder=outbox&idfrom=1&idto=999999999
```

RESPONSE:

Response: **OK**
Response (when delete operation was not successful): **Error**
Response (when wrong logindata): **Invalid login or password**

RESPONSE (XML):

Response:
```
<xml>
  <status>ok</status>
</xml>
```

Response (when delete operation was not successful):
```
<xml>
  <status>error</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

14. Delete SMS: JSONRPC method

HTTP POST METHOD:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
folder	one of the following: inbox, outbox, sentitems
idfrom	minimal id of message
idto	maximal id of message
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
Delete message with id=1234 from inbox:
{"method":"sms.delete_sms",
"params":{"login":"john","pass":"doe","folder":"inbox","idfrom":"1234""
idto":"1234"}}

Delete messages with id 1234 - 1250 from inbox:
{"method":"sms.delete_sms",
"params":{"login":"john","pass":"doe","folder":"inbox","idfrom":"1234",
"idto":"1250"}}

Delete all messages from outbox:
{"method":"sms.delete_sms",
"params":{"login":"john","pass":"doe","folder":"outbox","idfrom":"1","i
dto":"999999999"}}
```

RESPONSE:

Response: `{"result": "OK"}`
Response (when delete operation was not successful): `{"result": "Error"}`
Response (when wrong logindata): `{"result": "Invalid login or password"}`

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when delete operation was not successful):
`{"result":{"status":"error"}}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`

15. Get outgoing queue length: HTTP GET method

HTTP GET METHOD:
https://url-of-smseagle/index.php/http_api/get_queue_length

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/get_queue_length?
login=john&pass=doe
```

RESPONSE:
Response: **[number of messages in database that wait to be processed by GSM-modem]**
Sample response: 7
Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):
Response:
```
<xml>
  <queue_length>
    [number of messages in database that wait to be processed by GSM-modem]
  </queue_length >
  <status>ok</status>
</xml>
```

Sample response:
```
<xml>
  <queue_length>7</queue_length >
  <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
  <error_text>Wrong parameters</error_text>
  <status>error</status>
</xml>
```

16. Get outgoing queue length: JSONRPC method

HTTP POST METHOD CALL:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

BODY:
```
{"method":"sms.get_queue_length",
"params":{"login":"john","pass":"doe"}}
```

RESPONSE:

Response: `{"result": [number of messages in database that wait to be processed by GSM-modem]}`

Sample response: `{"result":7}`

Response: `{"result": "Invalid login or password"}`

Response: `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):

Response:
```
{"result":{"queue_length":[number of messages in database that wait to
be processed by GSM-modem],"status":"ok"}}
```

Sample response: `{"result": {"queue_length":"419","status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong parameters","status":"error"}}
```

17. Get inbox length: HTTP GET method

HTTP GET METHOD:
`https://url-of-smseagle/index.php/http_api/get_inbox_length`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle

responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/get_inbox_length?
login=john&pass=doe
```

RESPONSE:

Response: **[number of messages in database Inbox folder]**

Sample response: 3

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):

Response:
```
<xml>
   <queue_length>
      [number of messages in database Inbox folder]
   </queue_length>
   <status>ok</status>
</xml>
```

Sample response:
```
<xml>
   <inbox_length>3</inbox_length>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong parameters</error_text>
   <status>error</status>
</xml>
```

18. Get inbox length: JSONRPC method

HTTP POST METHOD:

```
https://url-of-smseagle/index.php/jsonrpc/sms
```

Parameter	Description
login	your user to login to SMSEagle

pass	your password to login to SMSEagle
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:
```
{"method":"sms.get_inbox_length",
"params":{"login":"john","pass":"doe"}}
```

RESPONSE:

Response: `{"result": "[number of messages in database Inbox folder]"}`

Sample response: 3

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters): `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):

Response:
```
{"result":{"inbox_length":[number of messages in database Inbox
folder],"status":"ok"}}
```

Sample response: `{"result": {"inbox_length":"3","status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong parameters","status":"error"}}
```

19. Get sentitems length: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/get_inbox_length
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/get_sentitems_length?
login=john&pass=doe
```

RESPONSE:

Response: **[number of messages in database Sentitems folder]**
Sample response: 21

Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):

Response:
```
<xml>
   <sentitems_length>
      [number of messages in database Inbox folder]
   </sentitems_length>
   <status>ok</status>
</xml>
```

Sample response:
```
<xml>
   <sentitems_length>21</sentitems_length>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong parameters</error_text>
   <status>error</status>
</xml>
```

20. Get sentitems length: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"sms.get_sentitems_length",
"params":{"login":"john","pass":"doe"}}
```

RESPONSE:

Response: `{"result": "[number of messages in database Sentitems folder]"}`
Sample response: `{"result": "21"}`
Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when wrong parameters): `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):

Response:
```
{"result":{"sentitems_length":[number of messages in database Sentitems
folder],"status":"ok"}}
```

Sample response: `{"result": {"sentitems_length":"21","status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong parameters","status":"error"}}
```

21. Get GSM/3G signal strength: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/get_gsmsignal
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
modem_no	*(optional parameter)* modem number to be queried (default = 1). Used only in multimodem devices
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/get_gsmsignal?
login=john&pass=doe&modem_no=1
```

RESPONSE:

Response: **GSM/3G signal strength in percent (values between 0-100).** If 3G modem is disconnected from GSM/3G network, method returns -1
Sample response: 74
Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong parameters**

RESPONSE (XML):

Response:
```
<xml>
   <signal_strength>
     [GSM signal strength in percent (values between 0-100)]
   </signal_strength>
   <status>ok</status>
</xml>
```

Sample response:
```
<xml>
   <signal_strength>74</signal_strength>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong parameters</error_text>
   <status>error</status>
</xml>
```

22. Get GSM/3G signal strength: JSONRPC method

HTTP POST METHOD CALL:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
modem_no	*(optional parameter)* modem number to be queried (default = 1). Used only in multimodem devices
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

BODY:

```
{"method":"signal.get_gsmsignal",
"params":{"login":"john","pass":"doe"}}
```

RESPONSE:

Response: `{"result": GSM/3G signal strength in percent: values between 0-100. If 3G modem is disconnected from GSM/3G network, method returns -1 }`
Sample response: `{"result":7}`
Response: `{"result": "Invalid login or password"}`
Response: `{"result": "Wrong parameters"}`

RESPONSE (EXTENDED):

Response:
`{"result":{"signal_strength":[number of messages in database Sentitems folder],"status":"ok"}}`

Sample response: `{"result": {"signal_strength":"7","status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

23. Phonebook group create: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/group_create
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
groupname	name for the created group
public	*(optional parameter)* 0 = private group, 1 = public group
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/group_create?
login=john&pass=doe&groupname=myusers&public=1
```

RESPONSE:
Response: **OK; ID=[ID of created group]**
Sample response: OK; ID=5
Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong or missing >>groupname<< parameter**

RESPONSE (XML):
Response:
```
<xml>
  <group_id>[ID of created group]</group_id>
  <status>ok</status>
</xml>
```

Sample response:
```
<xml>
  <group_id>5</group_id>
  <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
  <error_text>Wrong or missing >>groupname<< parameter</error_text>
  <status>error</status>
</xml>
```

24. Phonebook group create: JSONRPC method

HTTP POST METHOD:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
groupname	name for the created group
public	*(optional parameter)* 0 = private group, 1 = public group
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:
```
{"method":"phonebook.group_create",
"params":{"login":"john","pass":"doe","groupname":"myusers","public":"1
"}}
```

RESPONSE:
Response: `{"result": "OK; ID=[ID of created group]"}`
Sample response: `{"result": "OK; ID=5"}`
Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when wrong parameters): `{"result": "Wrong or missing >>groupname<< parameter"}`

RESPONSE (EXTENDED):
Response:
`{"result": {"group_id":"[ID of created group]","status":"ok"}}`

Sample response: `{"result": {"group_id":"748","status":"ok"}}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`

Response (when wrong parameters):
`{"result": {"error_text":"Wrong parameters","status":"error"}}`

25. Phonebook group read: HTTP GET method

HTTP GET METHOD:
`https://url-of-smseagle/index.php/http_api/group_read`

Parameter	Description

login	your user to login to SMSEagle
pass	your password to login to SMSEagle
public	*(optional parameter)* 0 = private group (default value), 1 = public group
uid	*(optional parameter)* id of user who created the group
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/group_read?
login=john&pass=doe&public=1&uid=12
```

RESPONSE:

Sample response: link

Response (when no data): **No data to display**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>uid<< parameter

Wrong or missing >>public<< parameter

RESPONSE (XML):

Sample response:

```
<xml>
 <groups>
  <item>
   <Name>private</Name>
   <ID>2</ID>
   <id_user>2</id_user>
   <is_public>true</is_public>
  </item>
  <item>
   <Name>Everyone</Name>
   <ID>3</ID>
   <id_user>1</id_user>
   <is_public>true</is_public>
  </item>
  <item>
   <Name>work</Name>
   <ID>4</ID>
   <id_user>1</id_user>
   <is_public>true</is_public>
  </item></groups>
 <status>ok</status>
</xml>
```

Response (when no data):

```
<xml>
    <error_text>No data to display</error_text>
```

```
   <status>error</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
   <error_text>Wrong or missing >>uid<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
   <error_text>Wrong or missing >>public<< parameter</error_text>
   <status>error</status>
</xml>
```

26. Phonebook group read: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
public	*(optional parameter)* 0 = private group (default value), 1 = public group
uid	*(optional parameter)* id of user who created the group
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"phonebook.group_read",
"params":{"login":"john","pass":"doe","public":"1","uid":"12"}}
```

RESPONSE:

Sample response:

```
{"result":[
 {"Name":"private","ID":"2","id_user":"1","is_public":"true"},
 {"Name":"Everyone","ID":"3","id_user":"1","is_public":"true"},
 {"Name":"work","ID":"4","id_user":"2","is_public":"true"}
]}
```

Response (when no data): `{"result": "No data to display"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):

```
{"result": "Wrong or missing >>uid<< parameter"}
{"result": "Wrong or missing >>public<< parameter"}
```

RESPONSE (EXTENDED):

Sample response:

```
{"result":[{"groups":[
 {"Name":"private","ID":"2","id_user":"1","is_public":"true"},
 {"Name":"Everyone","ID":"3","id_user":"1","is_public":"true"},
 {"Name":"work","ID":"4","id_user":"2","is_public":"true"}
],"status":"ok"}}
```

Response (when no data):

```
{"result": {"error_text":" No data to display","status":"error"}}
```

Response (when wrong logindata):

```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):

```
{"result": {"error_text":"Wrong or missing >>uid<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>public<<
parameter","status":"error"}}
```

27. Phonebook group update: HTTP GET method

HTTP GET METHOD:

```
https://url-of-smseagle/index.php/http_api/group_update
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	id of existing group
groupname	name for the group
public	*(optional parameter)* 0 = private group, 1 = public group
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/group_update?
login=john&pass=doe&group_id=2&groupname=myusers&public=1
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters):
Wrong or missing >>groupname<< parameter
Wrong or missing >>group_id<< parameter
Response (when group_id is wrong): **Group with the given id does not exists**

RESPONSE (XML):

Response:
```
<xml>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong or missing >>groupname<< parameter</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>group_id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when group_id is wrong):
```
<xml>
   <error_text>Group with the given id does not exists</error_text>
   <status>error</status>
</xml>
```

28. Phonebook group update: JSONRPC method

HTTP POST METHOD:

```
https://url-of-smseagle/index.php/jsonrpc/sms
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	id of existing group
groupname	name for the group
public	*(optional parameter)* 0 = private group, 1 = public group

responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:
```
{"method":"phonebook.group_update",
"params":{"login":"john","pass":"doe","group_id":"2","groupname":"myuse
rs","public":"1"}}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):
```
{"result": "Wrong or missing >>groupname<< parameter"}
{"result": "Wrong or missing >>group_id<< parameter"}
```
Response (when group_id is wrong): `{"result": "Group with the given id does not exists"}`

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>groupname<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>group_id<<
parameter","status":"error"}}
```

Response (when group_id is wrong):
```
{"result": {"error_text":"Group with the given id does not
exists","status":"error"}}
```

29. Phonebook group delete: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/group_delete
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	id of existing group
groupname	name of existing group
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/group_delete?
login=john&pass=doe&group_id=2&groupname=myusers
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>groupname<< parameter

Wrong or missing >>group_id<< parameter

Response (when group_id is wrong): **Group with the given id and name does not exist**

RESPONSE (XML):

Response:
```
<xml>
  <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
  <error_text>Wrong or missing >>groupname<< parameter</error_text>
  <status>error</status>
</xml>

<xml>
  <error_text>Wrong or missing >>group_id<< parameter</error_text>
  <status>error</status>
</xml>
```

Response (when group_id is wrong):
```
<xml>
  <error_text>Group with the given id does not exists</error_text>
  <status>error</status>
</xml>
```

30. Phonebook group delete: JSONRPC method

HTTP POST METHOD:

```
https://url-of-smseagle/index.php/jsonrpc/sms
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle

group_id	id of existing group
groupname	name of existing group
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"phonebook.group_delete",
"params":{"login":"john","pass":"doe","group_id":"2","groupname":"myuse
rs"}}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):

```
{"result": "Wrong or missing >>groupname<< parameter"}
{"result": "Wrong or missing >>group_id<< parameter"}
```

Response (when group_id is wrong): `{"result": "Group with the given id and name does not exist"}`

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):

```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):

```
{"result": {"error_text":"Wrong or missing >>groupname<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>group_id<<
parameter","status":"error"}}
```

Response (when group_id is wrong):

```
{"result": {"error_text":"Group with the given id does not
exists","status":"error"}}
```

31. Phonebook group add contact: HTTP GET method

HTTP GET METHOD:

```
https://url-of-smseagle/index.php/http_api/group_addcontact
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	id of existing group (or id's separated with comma)
contact_id	id of contact. The contact will be added to the group

responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/group_addcontact?
login=john&pass=doe&group_id=2&contact_id=1
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>group_id<< parameter

Wrong or missing >>contact_id<< parameter

Response (when id is wrong):

Group with the given id does not exists

Contact with the given id does not exists

RESPONSE (XML):

Response:

```
<xml>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
   <error_text>Wrong or missing >>group_id<< parameter</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>contact_id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when id is wrong):

```
<xml>
   <error_text>Group with the given id does not exists</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Contact with the given id does not exists</error_text>
   <status>error</status>
</xml>
```

32. Phonebook group add contact: JSONRPC method

HTTP POST METHOD:

```
https://url-of-smseagle/index.php/jsonrpc/sms
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	id of existing group (or id's separated with comma)
contact_id	id of contact. The contact will be added to the group
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"phonebook.group_addcontact",
"params":{"login":"john","pass":"doe","group_id":"2","contact_id":"1"}}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):

```
{"result": "Wrong or missing >>group_id<< parameter"}
{"result": "Wrong or missing >>contact_id<< parameter"}
```

Response (when id is wrong):

```
{"result": "Group with the given id does not exists"}
{"result": "Contact with the given id does not exists"}
```

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):

```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):

```
{"result": {"error_text":"Wrong or missing >>group_id<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>contact_id<<
parameter","status":"error"}}
```

Response (when id is wrong):

```
{"result": {"error_text":"Group with the given id does not
exists","status":"error"}}

{"result": {"error_text":"Contact with the given id does not
exists","status":"error"}}
```

33. Phonebook group remove contact: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/group_removecontact
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	id of existing group (or id’s separated with comma)
contact_id	id of contact. The contact will be removed from the group
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/group_removecontact?
login=john&pass=doe&group_id=2&contact_id=1
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>group_id<< parameter

Wrong or missing >>contact_id<< parameter

Response (when id is wrong):

Group with the given id does not exists

Contact with the given id does not exists

RESPONSE (XML):

Response:
```
<xml>
  <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
  <error_text>Wrong or missing >>group_id<< parameter</error_text>
  <status>error</status>
</xml>
```

```
<xml>
   <error_text>Wrong or missing >>contact_id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when id is wrong):

```
<xml>
   <error_text>Group with the given id does not exists</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Contact with the given id does not exists</error_text>
   <status>error</status>
</xml>
```

34. Phonebook group remove contact: JSONRPC method

HTTP POST METHOD:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	id of existing group (or id's separated with comma)
contact_id	id of contact. The contact will be removed from the group
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"phonebook.group_removecontact",
"params":{"login":"john","pass":"doe","group_id":"2","contact_id":"1"}}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):

```
{"result": "Wrong or missing >>group_id<< parameter"}
{"result": "Wrong or missing >>contact_id<< parameter"}
```

Response (when id is wrong):

```
{"result": "Group with the given id does not exists"}
{"result": "Contact with the given id does not exists"}
```

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):

```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>group_id<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>contact_id<<
parameter","status":"error"}}
```

Response (when id is wrong):
```
{"result": {"error_text":"Group with the given id does not
exists","status":"error"}}

{"result": {"error_text":"Contact with the given id does not
exists","status":"error"}}
```

35. Phonebook contact create: HTTP GET method

HTTP GET METHOD:

```
https://url-of-smseagle/index.php/http_api/contact_create
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
contactname	name for the created contact
number	telephone number for the created contact
public	*(optional parameter)* 0 = private contact, 1 = public contact (default value)
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/contact_create?
login=john&pass=doe&contactname=johndoe&number=12345678&public=1
```

RESPONSE:
Response: **OK; ID=[ID of created contact]**

Sample response: OK; ID=2

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>contactname<< parameter

Wrong or missing >>number<< parameter

RESPONSE (XML):
Response:
```
<xml>
  <contact_id>[ID of created contact]</contact_id>
  <status>ok</status>
```

</xml>

Sample response:

```
<xml>
   <contact_id>2</contact_id>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
   <error_text>Wrong or missing >>contactname<< parameter</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>number<< parameter</error_text>
   <status>error</status>
</xml>
```

36. Phonebook contact create: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
contactname	name for the created contact
number	telephone number for the created contact
public	*(optional parameter)* 0 = private contact 1 = public contact (default value)
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"phonebook.contact_create",
"params":{"login":"john","pass":"doe","contactname":"johndoe","number":
"12345678","public":"1"}}
```

RESPONSE:

Response: `{"result": "OK; ID=[ID of created contact]"}`

Sample response: `{"result": "OK; ID=2"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):
```
{"result": "Wrong or missing >>contactname<< parameter"}
{"result": "Wrong or missing >>number<< parameter"}
```

RESPONSE (EXTENDED):

Response:
```
{"result": {"contact_id":"[ID of created contact]","status":"ok"}}
```

Sample response: `{"result": {"contact_id":"2","status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>contactname<<
parameter"},"status":"error"}}

{"result": {"error_text":"Wrong or missing >>number<<
parameter","status":"error"}}
```

37. Phonebook contact read: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/contact_read
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
public	*(optional parameter)* 0 = private contacts (default value), 1 = public contacts
uid	*(optional parameter)* id of user who created the contact
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/contact_read?
login=john&pass=doe&public=1&uid=12
```

RESPONSE:

Sample response: link

Response (when no data): **No data to display**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>uid<< parameter

Wrong or missing >>public<< parameter

RESPONSE (XML):

Sample response:

```
<xml>
 <contacts>
  <item>
   <ID>2</ID>
   <GroupID>-1</GroupID>
   <Name>John Doe</Name>
   <Number>123123123</Number>
   <id_user>1</id_user>
   <is_public>true</is_public>
  </item>
  <item>
   <ID>4</ID>
   <GroupID>-1</GroupID>
   <Name>Jan Nowak</Name>
   <Number>4215456456</Number>
   <id_user>1</id_user>
   <is_public>true</is_public>
  </item>
  <item>
   <ID>5</ID>
   <GroupID>-1</GroupID>
   <Name>Andy</Name>
   <Number>+441234155931</Number>
   <id_user>1</id_user>
   <is_public>true</is_public>
  </item>
 </contacts>
 <status>ok</status>
</xml>
```

Response (when no data):

```
<xml>
    <error_text>No data to display</error_text>
    <status>error</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
    <error_text>Invalid login or password</error_text>
    <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
    <error_text>Wrong or missing >>uid<< parameter</error_text>
    <status>error</status>
</xml>

<xml>
    <error_text>Wrong or missing >>public<< parameter</error_text>
    <status>error</status>
</xml>
```

38. Phonebook contact read: JSONRPC method

HTTP POST METHOD:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
public	*(optional parameter)* 0 = private contacts (default value), 1 = public contacts
uid	*(optional parameter)* id of user who created the contact
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:
```
{"method":"phonebook.contact_read",
"params":{"login":"john","pass":"doe","public":"1","uid":"12"}}
```

RESPONSE:
Sample response:

```
{"result":[
 {"ID":"2","GroupID":"-1","Name":"John
Doe","Number":"123123123","id_user":"1","is_public":"false"},
 {"ID":"4","GroupID":"-1","Name":"Jan
Nowak","Number":"4215456456","id_user":"1","is_public":"false"},
 {"ID":"5","GroupID":"-
1","Name":"Andy","Number":"+441234155931","id_user":"1","is_public":"fa
lse"}
]}
```

Response (when no data): `{"result": "No data to display"}`
Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when wrong parameters):
```
{"result": "Wrong or missing >>uid<< parameter"}
{"result": "Wrong or missing >>public<< parameter"}
```

RESPONSE (EXTENDED):
Sample response:

```
{"result":{"contacts":[
 {"ID":"2","GroupID":"-1","Name":"John
Doe","Number":"123123123","id_user":"1","is_public":"false"},
 {"ID":"4","GroupID":"-1","Name":"Jan
Nowak","Number":"4215456456","id_user":"1","is_public":"false"},
 {"ID":"5","GroupID":"-
1","Name":"Andy","Number":"+441234155931","id_user":"1","is_public":"fa
lse"}
],"status":"ok"}}
```

Response (when no data):

```
{"result": {"error_text":" No data to display","status":"error"}}
```

Response (when wrong logindata):

```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):

```
{"result": {"error_text":"Wrong or missing >>uid<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>public<<
parameter","status":"error"}}
```

39. Phonebook contact update: HTTP GET method

HTTP GET METHOD:

```
https://url-of-smseagle/index.php/http_api/contact_update
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
contact_id	id of existing contact
contactname	name for the contact
number	phone number for the contact
public	*(optional parameter)* 0 = private group, 1 = public group
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/contact_update?
login=john&pass=doe&contact_id=4&contactname=johnlord&number=123456789&
public=1
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>contactname<< parameter

Wrong or missing >>contact_id<< parameter

Wrong or missing >>number<< parameter

Response (when contact_id is wrong): **Contact with the given id does not exists**

RESPONSE (XML):

Response:
```
<xml>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong or missing >>contactname<< parameter</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>contact_id<< parameter</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>number<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when contact_id is wrong):
```
<xml>
   <error_text>Contact with the given id does not exists</error_text>
   <status>error</status>
</xml>
```

40. Phonebook contact update: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
contact_id	id of existing contact
contactname	name for the contact
number	phone number for the contact
public	*(optional parameter)* 0 = private group, 1 = public group
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:

```
{"method":"phonebook.contact_update",
"params":{"login":"john","pass":"doe","contact_id":"4","contactname":"j
ohnlord","number":"123456789","public":"1"}}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):

```
{"result": "Wrong or missing >>contactname<< parameter"}
{"result": "Wrong or missing >>contact_id<< parameter"}
{"result": "Wrong or missing >>number<< parameter"}
```

Response (when contact_id is wrong): `{"result": "Contact with the given id does not exists"}`

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):

```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):

```
{"result": {"error_text":"Wrong or missing >>contactname<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>contact_id<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>number<<
parameter","status":"error"}}
```

Response (when contact_id is wrong):

```
{"result": {"error_text":"Contact with the given id does not
exists","status":"error"}}
```

41. Phonebook contact delete: HTTP GET method

HTTP GET METHOD:

`https://url-of-smseagle/index.php/http_api/contact_delete`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
contact_id	id of existing contact
contactname	name of existing contact

responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/contact_delete?
login=john&pass=doe&contact_id=4&contactname=johnlord
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>contactname<< parameter

Wrong or missing >>contact_id<< parameter

Response (when contact_id is wrong): **Contact with the given id and name does not exists**

RESPONSE (XML):

Response:
```
<xml>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong or missing >>contactname<< parameter</error_text>
   <status>error</status>
</xml>
```

```
<xml>
   <error_text>Wrong or missing >>contact_id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when contact_id is wrong):
```
<xml>
   <error_text>Contact with the given id and name does not exists </error_text>
   <status>error</status>
</xml>
```

42. Phonebook contact delete: JSONRPC method

HTTP POST METHOD:
```
https://url-of-smseagle/index.php/jsonrpc/sms
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
contact_id	id of existing contact
contactname	name of existing contact
responsetype	*(optional parameter)* simple = format response as simple object with one result field (default), extended = format response as extended JSON object

EXAMPLES:
```
{"method":"phonebook.contact_delete",
"params":{"login":"john","pass":"doe","contact_id":"4","contactname":"j
ohnlord"}}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):

`{"result": "Wrong or missing >>contactname<< parameter"}`

`{"result": "Wrong or missing >>contact_id<< parameter"}`

Response (when contact_id is wrong): `{"result": "Contact with the given id and name does not exists"}`

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>contactname<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>contact_id<<
parameter","status":"error"}}
```

Response (when contact_id is wrong):
```
{"result": {"error_text":"Contact with the given id and name does not
exists","status":"error"}}
```

43. Phonebook shift create: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/shift_create
```

Parameter	Description

login	your user to login to SMSEagle
pass	your password to login to SMSEagle
name	name for the created shift
enabled	0 = disabled, 1 = enabled
(mon-sun)_from	*shift start hour for each day of week*
(mon-sun)_to	*shift end hour for each day of week*
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/shift_create?
login=john&pass=doe&name=myshift&mon_from=08:00&mon_to=16:00&wed_from=0
9:00&wed_to=20:00&enabled=1
```

RESPONSE:

Response: **OK; ID=[ID of created shift]**
Sample response: OK; ID=5
Response (when wrong logindata): **Invalid login or password**
Response (when wrong parameters): **Wrong or missing >>name<< parameter**

RESPONSE (XML):

Response:

```
<xml>
  <shift_id>[ID of created shift]</shift_id>
  <status>ok</status>
</xml>
```

Sample response:

```
<xml>
  <shift_id>5</shift_id>
  <status>ok</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
  <error_text>Wrong or missing >>name<< parameter</error_text>
  <status>error</status>
</xml>
```

44. Phonebook shift create: JSONRPC method

HTTP POST METHOD:

```
https://url-of-smseagle/index.php/jsonrpc/sms
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
name	name for the created shift
enabled	0 = disabled, 1 = enabled
(mon-sun)_from	*shift start hour for each day of week*
(mon-sun)_to	*shift end hour for each day of week*
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
{"method":"phonebook.shift_create",
"params":{"login":"john","pass":"doe","name":"myshift","mon_from":"08:0
0","mon_to":"16:00","wed_from":"09:00","wed_to":"20:00","enabled":"1"}}
```

RESPONSE:
Response: `{"result": "OK; ID=[ID of created shift]"}`
Sample response: `{"result": "OK; ID=5"}`
Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when wrong parameters): `{"result": "Wrong or missing >>name<< parameter"}`

RESPONSE (EXTENDED):
Response:
`{"result": {"shift_id":"[ID of created shift]","status":"ok"}}`

Sample response: `{"result": {"shift_id":"748","status":"ok"}}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`

Response (when wrong parameters):
`{"result": {"error_text":" Wrong or missing >>name<< parameter ","status":"error"}}`

45. Phonebook shift read: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/shift_read
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
name	*(optional parameter)* shift name

enabled	*(optional parameter)* 0 = disabled, 1 = enabled
shift_id	*(optional parameter)* shift id
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/shift_read?
login=john&pass=doe&name=myshift
```

RESPONSE:

Sample response: link

Response (when no data): **No data to display**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong >>shift_id<< parameter

RESPONSE (XML):

Sample response:

```
<xml>
 <shifts>
  <shift>
   <id_shift>62</id_shift>
   <name>myshift</name>
   <mon_from>08:00</mon_from>
   <mon_to>16:00</mon_to>
   <tue_from/>
   <tue_to/>
   <wed_from>09:00</wed_from>
   <wed_to>20:00</wed_to>
   <thu_from/>
   <thu_to/>
   <fri_from/>
   <fri_to/>
   <sat_from/>
   <sat_to/>
   <sun_from/>
   <sun_to/>
   <enabled>true</enabled>
  </shift>
 </shifts>
 <status>ok</status>
</xml>
```

Response (when no data):

```
<xml>
  <error_text>No data to display</error_text>
  <status>error</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
```

```
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
   <error_text> Wrong >>id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):

```
<xml>
   <error_text> Wrong >>enabled<< parameter</error_text>
   <status>error</status>
</xml>
```

46. Phonebook shift read: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
name	*(optional parameter)* shift name
enabled	*(optional parameter)* 0 = disabled, 1 = enabled
shift_id	*(optional parameter)* shift id
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
{"method":"phonebook.shift_read",
"params":{"login":"john","pass":"doe","name":"myshift"}}
```

RESPONSE:

Sample response:

```
{
  "result": [
    {
      "id_shift": "62",
      "name": "myshift",
      "mon_from": "08:00",
      "mon_to": "16:00",
      "tue_from": null,
      "tue_to": null,
      "wed_from": "09:00",
      "wed_to": "20:00",
      "thu_from": null,
      "thu_to": null,
      "fri_from": null,
      "fri_to": null,
      "sat_from": null,
      "sat_to": null,
      "sun_from": null,
      "sun_to": null,
      "enabled": "true"
    }
  ]
}
```
Response (when no data): `{"result": "No data to display"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):
```
{"result": "Wrong >> enabled<< parameter"}
{"result": "Wrong >>shift_id<< parameter"}
```

RESPONSE (EXTENDED):

Sample response:
```
{
  "result": {
    "shifts": [
      {
        "id_shift": "62",
        "name": "myshift",
        "mon_from": "08:00",
        "mon_to": "16:00",
        "tue_from": null,
        "tue_to": null,
        "wed_from": "09:00",
        "wed_to": "20:00",
        "thu_from": null,
        "thu_to": null,
        "fri_from": null,
        "fri_to": null,
        "sat_from": null,
        "sat_to": null,
        "sun_from": null,
        "sun_to": null,
        "enabled": "false"
      }
    ],
    "status": "ok"
  }
}
```

Response (when no data):
```
{"result": {"error_text":" No data to display","status":"error"}}
```

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":" Wrong >>enabled<< parameter
","status":"error"}}

{"result": {"error_text":" Wrong >>shift_id<< parameter
","status":"error"}}
```

47. Phonebook shift update: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/shift_update
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
shift_id	id of existing shift
name	name for the shift
enabled	0 = disabled, 1 = enabled
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/shift_update?
login=john&pass=doe&shift_id=24&name=updatedshift&enabled=1
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>name<< parameter

Wrong or missing >>shift_id<< parameter

Response (when shift_id is wrong): **Shift with the given id does not exists**

RESPONSE (XML):

Response:
```
<xml>
  <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
```

```
</xml>
```

Response (when wrong parameters):

```
<xml>
   <error_text>Wrong or missing >>name<< parameter</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>shift_id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when shift_id is wrong):

```
<xml>
   <error_text>Shift with given id does not exists</error_text>
   <status>error</status>
</xml>
```

48. Phonebook shift update: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
shift_id	id of existing shift
name	name for the shift
enabled	0 = disabled, 1 = enabled
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
{"method":"phonebook.shift_update",
"params":{"login":"john","pass":"doe","shift_id":"24","name":"updatedsh
ift","enabled":"1"}}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):

```
{"result": "Wrong or missing >>name<< parameter"}
{"result": "Wrong or missing >>shift_id<< parameter"}
```

Response (when shift_id is wrong): `{"result": "Shift with the given id does not exists"}`

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>name<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>shift_id<<
parameter","status":"error"}}
```

Response (when shift_id is wrong):
```
{"result": {"error_text":"Shift with the given id does not
exists","status":"error"}}
```

49. Phonebook shift delete: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/shift_delete
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
shift_id	id of existing shift
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/shift_delete?
login=john&pass=doe&shift_id=24
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>shift_id<< parameter

Response (when shift_id is wrong): **Shift with the given id does not exist**

RESPONSE (XML):

Response:
```
<xml>
  <status>ok</status>
</xml>
```

Response (when wrong logindata):

```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>shift_id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when shift_id is wrong):

```
<xml>
   <error_text>Shift with the given id does not exists</error_text>
   <status>error</status>
</xml>
```

50. Phonebook shift delete: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
shift_id	id of existing shift
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
{"method":"phonebook.shift_delete",
"params":{"login":"john","pass":"doe","shift_id":"24"}}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):
`{"result": "Wrong or missing >>shift_id<< parameter"}`

Response (when shift_id is wrong): `{"result": "Shift with the given id does not exist"}`

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>shift_id<<
parameter","status":"error"}}
```

Response (when shift_id is wrong):
```
{"result": {"error_text":"Shift with the given id does not
exists","status":"error"}}
```

51. Phonebook shift add contact: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/shift_addcontact
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
shift_id	id of existing shift (or id's separated with comma)
contact_id	id of contact. The contact will be added to the shift
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/shift_addcontact?
login=john&pass=doe&shift_id=2&contact_id=1
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>shift_id<< parameter

Wrong or missing >>contact_id<< parameter

Response (when id is wrong):

Shift with the given id does not exists

Contact with the given id does not exists

RESPONSE (XML):

Response:
```
<xml>
  <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong or missing >>shift_id<< parameter</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>contact_id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when id is wrong):
```
<xml>
   <error_text>Shift with the given id does not exists</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Contact with the given id does not exists</error_text>
   <status>error</status>
</xml>
```

52. Phonebook shift add contact: JSONRPC method

HTTP POST METHOD:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
shift_id	id of existing shift (or id's separated with comma)
contact_id	id of contact. The contact will be added to the shift
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
{"method":"phonebook.shift_addcontact",
"params":{"login":"john","pass":"doe","shift_id":"24","contact_id":"1"}
}
```

RESPONSE:
Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):
```
{"result": "Wrong or missing >>shift_id<< parameter"}
{"result": "Wrong or missing >>contact_id<< parameter"}
```
Response (when id is wrong):

```
{"result": "Shift with the given id does not exists"}
{"result": "Contact with the given id does not exists"}
```

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>shift_id<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>contact_id<<
parameter","status":"error"}}
```

Response (when id is wrong):
```
{"result": {"error_text":"Shift with the given id does not
exists","status":"error"}}

{"result": {"error_text":"Contact with the given id does not
exists","status":"error"}}
```

53. Phonebook shift remove contact: HTTP GET method

HTTP GET METHOD:

```
https://url-of-smseagle/index.php/http_api/shift_removecontact
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
shift_id	id of existing shift (or id's separated with comma)
contact_id	id of contact. The contact will be added to the shift
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/shift_removecontact?
login=john&pass=doe&shift_id=24&contact_id=1
```

RESPONSE:

Response: **OK**

Response (when wrong logindata): **Invalid login or password**

Response (when wrong parameters):

Wrong or missing >>shift_id<< parameter

Wrong or missing >>contact_id<< parameter

Response (when id is wrong):

Shift with the given id does not exists
Contact with the given id does not exists

RESPONSE (XML):

Response:
```
<xml>
   <status>ok</status>
</xml>
```

Response (when wrong logindata):
```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when wrong parameters):
```
<xml>
   <error_text>Wrong or missing >>shift_id<< parameter</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Wrong or missing >>contact_id<< parameter</error_text>
   <status>error</status>
</xml>
```

Response (when id is wrong):
```
<xml>
   <error_text>Shift with the given id does not exists</error_text>
   <status>error</status>
</xml>

<xml>
   <error_text>Contact with the given id does not exists</error_text>
   <status>error</status>
</xml>
```

54. Phonebook shift remove contact: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
shift_id	id of existing shift (or id's separated with comma)
contact_id	id of contact. The contact will be added to the shift
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
{"method":"phonebook.shift_removecontact",
"params":{"login":"john","pass":"doe","shift_id":"24","contact_id":"1"}
}
```

RESPONSE:

Response: `{"result": "OK"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when wrong parameters):
```
{"result": "Wrong or missing >>shift_id<< parameter"}
{"result": "Wrong or missing >>contact_id<< parameter"}
```
Response (when id is wrong):
```
{"result": "Shift with the given id does not exists"}
{"result": "Contact with the given id does not exists"}
```

RESPONSE (EXTENDED):

Response: `{"result":{"status":"ok"}}`

Response (when wrong logindata):
```
{"result": {"error_text":"Invalid login or password","status":"error"}}
```

Response (when wrong parameters):
```
{"result": {"error_text":"Wrong or missing >>shift_id<<
parameter","status":"error"}}

{"result": {"error_text":"Wrong or missing >>contact_id<<
parameter","status":"error"}}
```

Response (when id is wrong):
```
{"result": {"error_text":"Shift with the given id does not
exists","status":"error"}}

{"result": {"error_text":"Contact with the given id does not
exists","status":"error"}}
```

55. Get modem state: HTTP GET method

HTTP GET METHOD:
```
https://url-of-smseagle/index.php/http_api/get_modem_state
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
modem_no	*(optional parameter)* modem number to be queried (default = 1). Used only in multimodem devices
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/get_modem_state?
login=john&pass=doe&modem_no=1
```

RESPONSE:

Response: **enabled / disabled**
Sample response: enabled
Response (when wrong logindata): **Invalid login or password**
Response (when modem doesn’t exist): **Wrong modem number**

RESPONSE (XML):

Response:

```
<xml>
 <modem_status>
   enabled / disabled
 </modem_status>
 <status>
   ok
 </status>
</xml>
```

Sample response:

```
<xml>
 <modem_status>
   enabled
 </modem_status>
 <status>
   ok
 </status>
</xml>
```

Response (when wrong logindata):

```
<xml>
   <error_text>Invalid login or password</error_text>
   <status>error</status>
</xml>
```

Response (when modem doesn’t exist):

```
<xml>
   <error_text> Wrong modem number</error_text>
   <status>error</status>
</xml>
```

56. Get modem state: JSONRPC method

HTTP POST METHOD:

```
https://url-of-smseagle/index.php/jsonrpc/sms
```

Parameter	Description
login	your user to login to SMSEagle

pass	your password to login to SMSEagle
modem_no	*(optional parameter)* modem number to be queried (default = 1). Used only in multimodem devices
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

BODY:
```
{"method":"sms.get_modem_state",
"params":{"login":"john","pass":"doe"}}
```

RESPONSE:
`{"result": enabled / disabled }`
Sample response: `{"result":"enabled"}`
Response (when wrong logindata): `{"result": "Invalid login or password"}`
Response (when modem doesn't exist): `{"result": "Wrong modem number"}`

RESPONSE (EXTENDED):
Response:
`{"result":{"modem_status": enabled / disabled,"status":"ok"}}`

Sample response: `{"result": {"modem_status":"Wrong modem number","status":"ok"}}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`
Response (when modem doesn't exist):
`{"result": {"error_text":"Wrong modem number","status":"error"}}`

57. Set modem state: HTTP GET method

HTTP GET METHOD:
`https://url-of-smseagle/index.php/http_api/set_modem_state`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
modem_no	*(optional parameter)* modem number for status change (default = 1). Used only in multimodem devices
status	*enabled / disabled*
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/set_modem_state?
login=john&pass=doe&modem_no=1&status=enabled
```

RESPONSE:
Response: **OK**
Response (when wrong logindata): **Invalid login or password**
Response (when modem doesn't exist): **Wrong modem number**
Response (when wrong modem state): **Wrong modem state**

RESPONSE (XML):
Response:
```
<xml>
 <status>
  ok
 </status>
</xml>
```

Response (when wrong logindata):
```
<xml>
  <error_text>Invalid login or password</error_text>
  <status>error</status>
</xml>
```

Response (when modem doesn't exist):
```
<xml>
  <error_text> Wrong modem number</error_text>
  <status>error</status>
</xml>
```

Response (when wrong modem state):
```
<xml>
  <error_text>Wrong modem state</error_text>
  <status>error</status>
</xml>
```

58. Set modem state: JSONRPC method

HTTP POST METHOD:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
modem_no	*(optional parameter)* modem number for status change (default = 1). Used only in multimodem devices
status	*enabled / disabled*
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

BODY:

```
{"method":"sms.set_modem_state", "params":{"login":"john","pass":"doe",
"status":"enabled"}}
```

RESPONSE:

`{"result": enabled / disabled }`

Sample response: `{"result":"enabled"}`

Response (when wrong logindata): `{"result": "Invalid login or password"}`

Response (when modem doesn't exist): `{"result": "Wrong modem number"}`

Response (when wrong modem state): `{"result": "Wrong modem state"}`

RESPONSE (EXTENDED):

Response:

`{"result":{"modem_status": enabled / disabled,"status":"ok"}}`

Sample response: `{"result": {"modem_status":"Wrong modem number","status":"ok"}}`

Response (when wrong logindata):

`{"result": {"error_text":"Invalid login or password","status":"error"}}`

Response (when modem doesn't exist):

`{"result": {"error_text":"Wrong modem number","status":"error"}}`

Response (when wrong modem state):

`{"result": {"error_text":"Wrong modem state","status":"error"}}`

59. User ID read: HTTP GET method

HTTP GET METHOD:

```
https://url-of-smseagle/index.php/http_api/userid_read
```

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
username	*username to be queried*
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:

```
https://url-of-smseagle/index.php/http_api/userid_read?
login=john&pass=doe&username=myuser
```

RESPONSE:

Response: **User ID**

Sample response: 24

Response (when username parameter is missing): **Missing >>username<< parameter**

Response (when user doesn't exist): **Wrong >>username<< parameter**

RESPONSE (XML):

Response:
```
<xml>
 <status>
  ok
 </status>
</xml>
```

Response (when username parameter is missing):
```
<xml>
  <error_text>Missing >>username<< parameter</error_text>
  <status>error</status>
</xml>
```

Response (when user doesn't exist):
```
<xml>
  <error_text>Wrong >>username<< parameter</error_text>
  <status>error</status>
</xml>
```

60. User ID read: JSONRPC method

HTTP POST METHOD:
`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
username	*username to be queried*
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

BODY:
```
{"method":"phonebook.userid_read",
"params":{"login":"john","pass":"doe","username":"myuser"}}
```

RESPONSE:
`{"result": [user ID] }`
Sample response: `{"result":"24"}`
Response (when username parameter is missing): `{"result": "Missing >>username<< parameter"}`
Response (when user doesn't exist): `{"result": "Wrong >>username<< parameter"}`

RESPONSE (EXTENDED):
Response:
`{"result":{"uid": [user ID],"status":"ok"}}`

Sample response: `{"result": {"uid":"24","status":"ok"}}`

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`
Response (when username parameter is missing):
```
{"result": {"error_text":"Missing >>username<<
parameter","status":"error"}}
```
Response (when user doesn't exist):
```
{"result": {"error_text":"Wrong >>username<<
parameter","status":"error"}}
```

61. Group members read: HTTP GET method

HTTP GET METHOD:
`https://url-of-smseagle/index.php/http_api/group_members_read`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	ID of group to be queried
user_id	*(optional parameter) show only contacts created by user with given ID*
public	*(optional parameter) 0 = private, 1 = public*
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

EXAMPLES:
```
https://url-of-smseagle/index.php/http_api/group_members_read?
login=john&pass=doe&group_id=11
```

RESPONSE:
Sample response: link
Response (when wrong or missing group_id parameter): **Wrong or missing >>group_id<< parameter**
Response (when wrong user_id parameter): **Wrong >>user_id<< parameter**
Response (when wrong public parameter): **Wrong >>public<< parameter**
Response (when result set is empty): **No data to display**

RESPONSE (XML):
Response:
```
<xml>
 <contacts>
  <contact>
   <ID>17</ID>
   <Name>mycontact1</Name>
   <Number>23456</Number>
   <id_user>1</id_user>
   <is_public>true</is_public>
  </contact>
  <contact>
   <ID>24</ID>
   <Name>mycontact3</Name>
   <Number>12345</Number>
   <id_user>3</id_user>
   <is_public>false</is_public>
  </contact>
 </contacts>
</xml>
```

Response (when wrong or missing group_id parameter):

```
<xml>
  <error_text>Wrong or missing >>group_id<< parameter</error_text>
  <status>error</status>
</xml>
```

Response (when wrong user_id parameter):

```
<xml>
  <error_text>Wrong >>user_id<< parameter</error_text>
  <status>error</status>
</xml>
```

Response (when wrong public parameter):

```
<xml>
  <error_text> Wrong >>public<< parameter </error_text>
  <status>error</status>
</xml>
```

62. Group members read: JSONRPC method

HTTP POST METHOD:

`https://url-of-smseagle/index.php/jsonrpc/sms`

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
group_id	ID of group to be queried
user_id	*(optional parameter) show only contacts created by user with given ID*
public	*(optional parameter) 0 = private, 1 = public*
responsetype	*(optional parameter)* text = format response as text (default), xml = format response as XML object

BODY:

```
{"method":"phonebook.group_members_read",
"params":{"login":"john","pass":"doe","group_id":"11"}}
```

RESPONSE:

Sample response:

```
{
"result": [
      {"ID":"1706","Name":"mycontact1","Number":"23456",
       "id_user":"1","is_public":"true"},
      {"ID":"1693","Name":"mycontact3","Number":"12345",
       "id_user":"3","is_public":"false"}
 ]
}
```

Response (when wrong or missing group_id parameter): `{"result": "Wrong or missing >>group_id<< parameter"}`
Response (when wrong user_id parameter): `{"result": "Wrong >>user_id<< parameter"}`
Response (when wrong public parameter): `{"result": "Wrong >>public<< parameter"}`

RESPONSE (EXTENDED):

Sample response:

```
{
  "result": {
    "contacts": [
           {"ID":"1706","Name":"mycontact1","Number":"23456",
            "id_user":"1","is_public":"true"},
           {"ID":"1693","Name": "mycontact3","Number":"12345",
            "id_user":"3","is_public":"false"}
    ],
    "status": "ok"
  }
}
```

Response (when wrong logindata):
`{"result": {"error_text":"Invalid login or password","status":"error"}}`
Response (when wrong or missing group_id parameter):
`{"result": {"error_text":"Wrong or missing >>group_id<< parameter","status":"error"}}`
Response (when wrong user_id parameter):
`{"result": "Wrong >>user_id<< parameter"}`
Response (when wrong public parameter):
`{"result": "Wrong >>public<< parameter"}`

Plugins and integration manuals for NMS & Auth systems

SMSEagle has a number of ready-to-use plugins and integration manuals for an easy and quick integration of SMSEagle device with external software (Network Monitoring Systems, Authentication Systems and other). The list grows constantly and is published on SMSEagle website. For a complete and up-to-date list of plugins please go to: http://www.smseagle.eu/integration-plugins/

Connecting directly to SMSEagle database

SMSEagle’s database operates on PostgreSQL database engine. It is possible to connect to the database from external application using the following credentials:

POSTGRESQL DATABASE CREDENTIALS

Host: IP address of your SMSEagle
Database name: smseagle
User: smseagleuser
Password: postgreeagle

Injecting short SMS using SQL

The simplest example is short text message (limited to 160 chars):

```
INSERT INTO outbox (
  DestinationNumber,
  TextDecoded,
  CreatorID,
  Coding,
  Class,
  SenderID
) VALUES (
  '1234567',
  'This is a SQL test message',
  'Program',
  'Default_No_Compression',
   -1,
   'smseagle1'
);

INSERT INTO user_outbox (
  id_outbox,
  id_user
) SELECT CURRVAL(pg_get_serial_sequence('outbox','ID')), 1;
```

In the above example the message will belong to user with **id_user** 1 (default 'admin'). You can find id_user values for other users in table public."user".

Injecting long SMS using SQL

Inserting multipart messages is a bit more tricky, you need to construct also UDH header and store it hexadecimally written into UDH field. Unless you have a good reason to do this manually, use API.

For long text message, the UDH starts with 050003 followed by byte as a message reference (you can put any hex value there, but it should be different for each message, D3 in following example), byte for number of messages (02 in example, it should be unique for each message you send to same phone number) and byte for number of current message (01 for first message, 02 for second, etc.).

For example long text message of two parts could look like following:

```
INSERT INTO outbox (
    "DestinationNumber",
    "CreatorID",
    "MultiPart",
    "UDH",
    "TextDecoded",
    "Coding",
    "Class",
    "SenderID"
) VALUES (
    '1234567',
    'Program',
    'true',
    '050003D30201',
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
minim veniam, qui',
    'Default_No_Compression',
    -1,
    'smseagle1'
)

INSERT INTO outbox_multipart (
    "ID",
    "SequencePosition",
    "UDH",
    "TextDecoded",
    "Coding",
    "Class"
) SELECT
    CURRVAL(pg_get_serial_sequence('outbox','ID')),
    2,
    '050003D30202',
    's nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat.',
    'Default_No_Compression',
    -1;

INSERT INTO user_outbox (
  id_outbox,
  id_user
) SELECT
  CURRVAL(pg_get_serial_sequence('outbox','ID')),
  1;
```

Note: Adding UDH means that you have less space for text, in above example you can use only 153 characters in single message.

Database cleaning scripts

We have added some useful scripts which may be used to delete SMS messages from database through Linux CLI.
Scripts are located at following directory:
`/mnt/nand-user/scripts/`

- **`db_delete`** – script for deleting SMS from folders Inbox, SentItems older than provided date. Usage:
  `./db_delete YYYYMMDDhhmm`

- **`db_delete_7days`** – script for deleting SMS from folders Inbox, Sentitems older than 7 days. Usage:
  `./db_delete_7days`

- **`db_delete_allfolders`** – script for cleaning PostgreSQL database folders (Inbox, SentItems, Outbox). Specially designed to run periodically through *cron.* Usage:
  `./db_delete_allfolders`

- **`db_delete_select -`** script for deleting SMS from chosen databse folder (Inbox, Outbox, SentItems, Trash). Usage:
  `./db_delete_select {inbox|outbox|sentitems|trash}`

Adding script to system *cron* daemon

Due to security reasons *crontab* is being erased while rebooting the device. Thats why *normal* cron entry won't work. Please see file */mnt/mtd/rcs* - this script is being launched due to startup of the device. Here you have some our cron rules added. You just need to add your rule at the end of the file.

Example:
*cron_add db_clean " 0 0 1 * * /mnt/nand-user/scripts/db_delete_allfolders"*
which will run cleaning script every 1st day of month.

SNMP agent

"Simple Network Management Protocol (SNMP) is an Internet-standard protocol for managing devices on IP networks. It is used mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention" (source: Wikipedia).

SMSeagle device has a built-in Net-SNMP agent. The SNMP agent provides access to Linux Host MIB tree of the device, and additionally (using extension NET-SNMP-EXTEND-MIB) allows access to custom metrics specific to SMSEagle.

Available SNMP metrics that describe a state of a SMSEagle device are:

Metric name	Description	OID
GSM_Signal	Returns GSM signal strength in percent. Value range: 0-100. If modem is disconnected from GSM network GSM_Signal returns 0.	.1.3.6.1.4.1.8072.1.3.2.3.1.2.10.71.83.77.95.83.105.103.110.97.108
GSM_NetName1	Returns Network Name used on current SIMcard	.1.3.6.1.4.1.8072.1.3.2.3.1.2.12.71.83.77.95.78.101.116.78.97.109.101.49
FolderOutbox_Total	Returns number of SMS messages in Outbox folder (outgoing queue length)	.1.3.6.1.4.1.8072.1.3.2.3.1.2.17.70.111.108.100.101.114.73.110.98.111.120.95.84.111.116.97.108
FolderInbox_Total	Returns number of SMS messages in Inbox folder	.1.3.6.1.4.1.8072.1.3.2.3.1.2.17.70.111.108.100.101.114.73.110.98.111.120.95.84.111.116.97.108
FolderSent_Last24H	Returns number of SMS messages sent from the device within last 24 hours	.1.3.6.1.4.1.8072.1.3.2.3.1.2.18.70.111.108.100.101.114.83.101.110.116.95.76.97.115.116.50.52.72
FolderSent_Last1M	Returns number of SMS messages sent from the device within last month	.1.3.6.1.4.1.8072.1.3.2.3.1.2.17.70.111.108.100.101.114.83.101.110.116.95.76.97.115.116.49.77
FolderSent_Last24HSendErr	Returns number of SMS messages sent with error within last 24h. Error occurs when GSM modem cannot send SMS message or message is rejected by GSM carrier (mostly happens when a credit on pre-paid SIM card is over)	.1.3.6.1.4.1.8072.1.3.2.3.1.2.25.70.111.108.100.101.114.83.101.110.116.95.76.97.115.116.50.52.72.83.101.110.100.69.114.114

RESULT VALUES

- Using OID

Result values for each custom metric are available and can be fetched from OID given in table above.

- Using textual name

Alternatively result values for each custom metric can be fetched using textual names from OID tree under: NET-SNMP-EXTEND-MIB::nsExtendOutputFull."[METRIC NAME]"

For example:
Result value for parameter ***GSM_Signal****:*
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."GSM_Signal"

If your chosen SNMP tool cannot access NET-SNMP-EXTEND-MIB objects, you can download MIB definitions from: http://www.smseagle.eu/download/NET-SNMP-EXTEND-MIB.txt

READING RESULT VALUES

In order to test-read the parameter values from SNMP agent you can use any tools available for SNMP protocol (for example: NET-SNMP library for Linux or iReasoning MiB-Browser for Windows).

EXAMPLE OF READING **GSM_SIGNAL** VALUE USING NET-SNMP LIBRARY

a) Command for reading the result value:

```
snmpget -v 2c -c public localhost
.1.3.6.1.4.1.8072.1.3.2.3.1.2.10.71.83.77.95.83.105.103.110.97.108
```

Result:

```
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."GSM_Signal" = STRING: 54
```

Comment: GSM Signal strength value is 54%

EXAMPLE OF READING **GSM_NETNAME** VALUE USING NET-SNMP LIBRARY

a) Command for reading the result value:

```
snmpget -v 2c -c public localhost
.1.3.6.1.4.1.8072.1.3.2.3.1.2.11.71.83.77.95.78.101.116.78.97.109.101
```

Result:

```
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."GSM_NetName" = STRING: Plus
```

Comment: Currently used network is Plus

EXAMPLE OF READING **FOLDEROUTBOX_TOTAL** VALUE USING NET-SNMP LIBRARY (AND TEXTUAL NAME OF METRIC)

a) Command for reading the result value:

```
snmpget -v 2c -c public ip-of-smseagle 'NET-SNMP-EXTEND-
MIB::nsExtendOutputFull."FolderOutbox_Total"'
```

Result:

```
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."FolderOutbox_Total" = STRING: 0
```

Comment: Number of SMS messages waiting in outbox queue is 0

EXAMPLE OF READING **SYSTEMUPTIME** FROM LINUX HOST USING NET-SNMP LIBRARY

a) Command for reading the result value:

```
snmpget -v 2c -c public ip-of-smseagle system.sysUpTime.0
```

Result:

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (216622) 0:36:06.22
```

Comment: Linux system is up for 36 hours, 6.22 minutes

EXAMPLE OF BROWSING SMSEAGLE EXTENSION PARAMETERS IN MIB TREE USING NET-SNMP LIBRARY

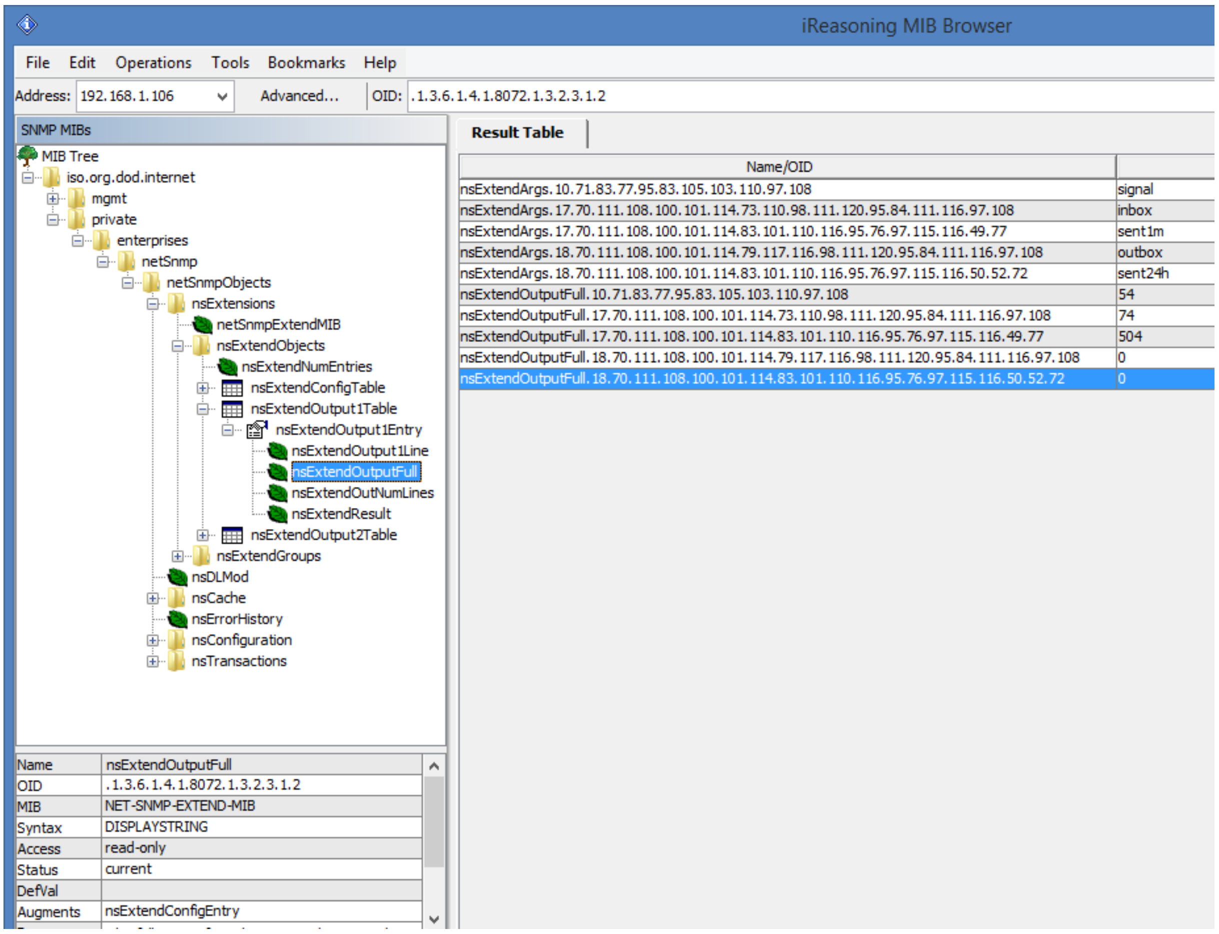
a) Command for reading the result value:

```
snmpwalk -v 2c -c public ip-of-smseagle .1.3.6.1.4.1.8072.1.3.2.3.1.2
```

Result:

```
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."GSM_Signal" = STRING: 54
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."GSM_NetName" = STRING: Plus
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."FolderInbox_Total" = STRING: 15
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."FolderSent_Last1M" = STRING: 19
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."FolderOutbox_Total" = STRING: 0
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."FolderSent_Last24H" = STRING: 0
NET-SNMP-EXTEND-MIB::nsExtendOutputFull." FolderSent_Last24HSendErr" = STRING: 0
```

EXAMPLE OF BROWSING SMSEAGLE EXTENSION PARAMETERS IN MIB TREE USING MIB-BROWSER



Name/OID	
nsExtendArgs.10.71.83.77.95.83.105.103.110.97.108	signal
nsExtendArgs.17.70.111.108.100.101.114.73.110.98.111.120.95.84.111.116.97.108	inbox
nsExtendArgs.17.70.111.108.100.101.114.83.101.110.116.95.76.97.115.116.49.77	sent1m
nsExtendArgs.18.70.111.108.100.101.114.79.117.116.98.111.120.95.84.111.116.97.108	outbox
nsExtendArgs.18.70.111.108.100.101.114.83.101.110.116.95.76.97.115.116.50.52.72	sent24h
nsExtendOutputFull.10.71.83.77.95.83.105.103.110.97.108	54
nsExtendOutputFull.17.70.111.108.100.101.114.73.110.98.111.120.95.84.111.116.97.108	74
nsExtendOutputFull.17.70.111.108.100.101.114.83.101.110.116.95.76.97.115.116.49.77	504
nsExtendOutputFull.18.70.111.108.100.101.114.79.117.116.98.111.120.95.84.111.116.97.108	0
nsExtendOutputFull.18.70.111.108.100.101.114.83.101.110.116.95.76.97.115.116.50.52.72	0

Name	nsExtendOutputFull
OID	.1.3.6.1.4.1.8072.1.3.2.3.1.2
MIB	NET-SNMP-EXTEND-MIB
Syntax	DISPLAYSTRING
Access	read-only
Status	current
DefVal	
Augments	nsExtendConfigEntry

Setting up SNMP v3 access control

By default SMSEagle devices uses SNMP v2 access control. Using v3 can strengthen security, however is not obsolete. To ease switch to SNMP v3 access control we've prepared special shell script located at */mnt/nand-user/smseagle* directory.

1. *Log in via SSH using root account*
2. *Navigate to:*
   cd /mnt/nand-user/smseagle/
3. *Configuration script:*
   ./snmpv3
4. *Script can run with following parameters:*
   i. *add*
   ii. *del*
   iii. *enablev2*
   iv. *disablev2*
5. *To add v3 USER please run:*
   ./snmpv3 add USERNAME PASSWORD ENCRYPTIONPASSWORD

6. *To delete USER please run:*
   ./snmpv3 del

7. *To disable v2 access policy run:*
   ./snmpv3 disablev2

8. *To enable v2 access policy run:*
   ./snmpv3 enablev2

Forwarding logs to external server

Our devices runs rsyslog for log managing. Here we describe how to configure additional rules for rsyslog daemon: rsyslogd. This is only a brief excerpt from rsyslog manual website. Full information is available at: http://www.rsyslog.com/

Rsyslogd configuration is managed using a configuration file located at */etc/rsyslog.conf*

- Forwarding all logs to external server (using TCP port)
  At the bottom of the configuration file add:
  ```
  *.*       @@server_ip_address:port
  ```
  eg.
  ```
  *.*       @@192.168.0.199:10514
  ```

- Forwarding all logs to external server (using UDP port)
  At the bottom of the configuration file add:
  ```
  *.*       @server_ip_address:port
  ```
  eg.
  ```
  *.*       @192.168.0.199:10514
  ```

- SSL-encryption of your log traffic: please have a look at this article:
  http://www.rsyslog.com/doc/v8-stable/tutorials/tls_cert_summary.html

Automatic software updates checks

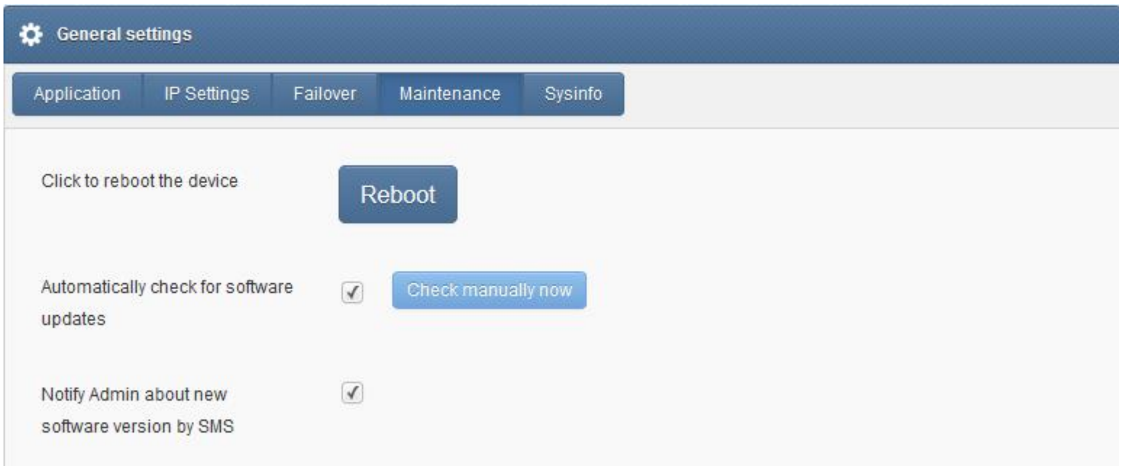
SMSEagle software is under process of continual improvement. We listen to our customers, and new releases are based on our customer's inputs/requests. Software updates are released frequently, and offer access to new features and fixes to reported issues. Web-GUI offers you a possibility to automatically check for new software updates. This can be done in two ways:

MANUAL CHECK

In order to manually check for available software updates go to menu Settings > tab Maintenance. Click on the button "Check manually now". At the top pops up a balloon in red with information if it is up-to-date.

AUTOMATIC CHECK

In order to start automatic checks for software updates go to menu Settings > tab Maintenance, and check the option "Automatically check for software updates". This will enable periodic checks (once a month) for available software updates. If a new update is available, a message "Update Available" will appear in menu Settings> Sysinfo – next to the current software version number.
If you select "Notify Admin about new software version by SMS", the device will additionally send SMS to the default admin account (if the phone number is entered in the account) with a notification about new software update.



Click to reboot the device

Automatically check for software updates ✓

Notify Admin about new software version by SMS ✓

Screenshot from "General settings-Maintenance"

Notice: Your SMSEagle device must have a HTTPS connectivity with address www.smseagle.eu in order for this feature to work.

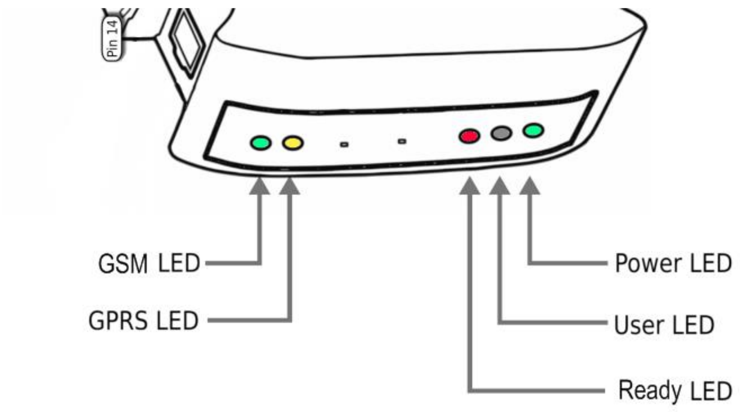
3. TROUBLESHOOTING

To make sure that the device is working properly, follow the three steps:

1. Verification of LEDs
2. Checking the device configuration (IP Settings)
3. Check the device logs (description below)

Verification of LEDs

Normal operation of the device is signaled by LEDs as follows:



LED	Correct operation
Power (PWR)	Continuously lit
User	Disabled
Ready (RDY)	Blinking
GSM	Slow flashing in stand-by mode, Quick flashing when modem in use
GPRS	Not used

Checking the device logs

SMSEagle operates on Linux system. Linux system log is available under menu position “Settings” > “Logs”. In case of any problems with the device this log is a valuable source of troubleshooting information.

Please attach information from this log when contacting with SMSEagle Support Team.

When the device is not reachable

1. Check if the device is correctly connected to the network. Check LED status of RJ45 socket.

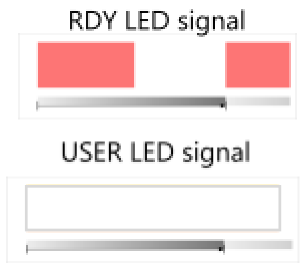
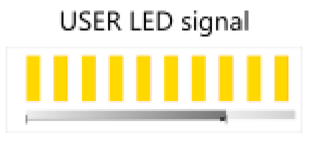
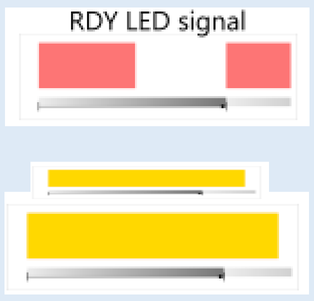
2. In the case when the device does not respond due to a malfunction or incorrect user settings please reboot the device by disconnecting and connecting power source (or pressing Reset switch).

3. If you still cannot connect with the device, it is possible to restore to factory IP settings by using the SW button.

Restoring factory defaults

This action restores the following settings to default values: **IP settings, database content**.

In order to restore factory defaults proceed with the following steps:

LED signaling	USER actions	System reaction
	**When the device is ready to operate:** When READY LED is blinking and USER LED is disabled (do not proceed if you don't have that state!)	
	1. **Press and hold SW button for 10 seconds**	Restore service is counting down.
	2. **Release SW button after 10 seconds.** User LED will begin to blink.	System is reading factory defaults. Factory settings are being applied to the device.
	3. **Wait until system reboots.** Default IP settings are restored.	System is going for a reboot.

Please note, that after reboot the device will be finishing the process of factory reset, therefore it can take longer for the system to start.

4. SERVICE AND REPAIR

Warranty

Your SMSEagle comes with 14 days of post-sales technical support (including assistance in integration with external software) and one year of hardware repair warranty coverage. For a detailed information on warranty terms and conditions check warranty card that comes with your device or follow the link: www.smseagle.eu/docs/general_warranty_terms_and_conditions.pdf

Service

Before contacting with support team, be sure that you have read Troubleshooting section of this manual. SMSEagle Support Team is available by email or telephone.

Support Email: support@smseagle.eu

Support telephone: + 48 61 6713 411

The support service is provided by:
Proximus Software
ul. Piątkowska 163,
60-650 Poznan, Poland

WHEN CONTACTING SUPPORT TEAM, BE PREPARED TO PROVIDE THE FOLLOWING INFORMATION:

System Information

To get information about your SMSEagle, go to menu Settings > Maintenance. You will find there information about application and database version.

System Logs

Go to menu Settings > Logs. If possible copy the log data and provide to support team when requested.

MAC address

Each SMSEagle device has its unique MAC address. MAC address is printed on the device body.

5. TECH SPECS AND SAFETY INFORMATION

Technical Specifications

Hardware Specification

- Processor type: ARM9 32bit 200MIPS RISC
- Network interface: Ethernet 10/100 TX (1xRJ45)
- 1GB Flashdisk
- RTC Clock: RTC 240B SRAM, Watchdog timer
- Power consumption: max 17W
- Noise level: Fanless
- Dimensions: (width x depth x height) 35 x 120 x 101 mm
- Weight: 350g
- Casing: ABS, DIN rail installation
- Operating parameters:
    - Operating temperature: 10 ~ 60°C
    - Humidity: 5 ~ 95% RH (no condensation)
- Modem GSM/GPRS:
    - Waveband: GSM/GPRS/EGPRS 900/1800/1900 MHz
    - Compatible with GSM phase 2/2+.
    - Class 4 (2W @ 900 MHz).
    - Class 1 (1W @1800/1900 MHz).
- SIM card standard: mini
- Antenna connector: SMA

Power Supply

AC line input

Voltage ranges: 100–240 V alternating current (AC)

Frequency: 50–60Hz single phase

GSM Antenna

Omnidirectional 3.5dBi antenna with magnetic foot

Cable length 3m

Sending/Receiving Throughput

- Incoming transmission rate: up to 30 SMS/min

- Outgoing transmission rate: up to 20 SMS/min
- API send SMS requests: 30 SMS/min (messages are queued for sending in a built-in database)

Software Platform

- Operating system: Linux 2.6
- built-in Apache2 web server
- built-in PostgreSQL database server
- built-in Postfix email server
- modern responsive web interface
- watchdog mechanisms for GSM/3G modem

Important Safety Information

This chapter provides important information about safety procedures. For your safety and that of your equipment, follow these rules for handling your device.

WARNING: Incorrect storage or use of your device may void the manufacturer's warranty. Failure to follow these safety instructions could result in fire, electric shock, or other injury or damage.

Always take the following precautions.

Disconnect the power plug from AC power source or if any of the following conditions exist:

- the power cord or plug becomes frayed or otherwise damaged
- you spill something into the case
- the device is exposed to rain or any other excess moisture
- the device has been dropped or the case has been otherwise damaged

Be sure about that the use of this product is allowed in your country and in the environment required. As with any other telecommunication equipment, the use of this product may be dangerous and has to be avoided in the following areas: where it can interfere with other electronic devices in environments such as hospitals, airports, aircrafts, etc.; where there is risk of explosion such as gasoline stations, oil refineries, etc. It is responsibility of the user to enforce the country regulation and the specific environment regulation.
Do not disassemble the product; any mark of tampering will compromise the warranty validity.

Every device has to be equipped with a proper antenna with specific characteristics. The antenna has to be installed with care in order to avoid any interference with other electronic devices and has to be installed with the guarantee of a minimum 20 cm distance from the body. In case of this requirement cannot be satisfied, the system integrator has to assess the final product against the SAR regulation.

DISCLAIMER: The manufacturer is not responsible for any damages caused by inappropriate installation, not maintaining the proper technical condition or using a product against its destination.