



# SMSEAGLE NXS-9750

User's Manual



---

# Congratulations on purchasing **SMSEAGLE**

The materials used in this publication are copyright and are not to be duplicated, copied, or used without the prior consent of the copyright holder. Technical specifications are subject to change without prior notice being given.

Document version: 3.2

# CONTENTS

What's In The Box .....	8
Prepare for First Start.....	9
Get to know with Connectors, Ports and LEDs.....	13
Basic Operations .....	14
SMSEagle basic features.....	14
Phonebook.....	15
Phonebook Contacts .....	15
Phonebook Groups .....	16
Phonebook Working Shifts.....	16
Reporting module.....	17
Statistics view .....	17
SMSEagle plugins.....	18
Autoreply plugin.....	18
Network Monitoring plugin.....	19
Email to SMS plugin .....	22
Email to SMS Poller.....	25
SMS to Email plugin .....	27
Callback URL plugin .....	28
SMS Forward .....	30
Periodic SMS.....	31
Digital input/output .....	32
Temperature & humidity sensor .....	35
Multimodem features.....	37
SMSEagle API.....	38
1. Send SMS: HTTP GET method.....	38
2. Send SMS: JSONRPC method .....	40
3. Send SMS to a group: HTTP GET method .....	41
4. Send SMS to a group: JSONRPC method.....	43
5. Send SMS to contact: HTTP GET method .....	44

6. Send SMS to contact: JSONRPC method.....	46
7. Send USSD code: HTTP GET method.....	47
8. Send USSD code: JSONRPC method.....	48
9. Send binary SMS: HTTP GET method.....	49
10. Send binary SMS: JSONRPC method.....	51
11. Read SMS: HTTP GET method.....	52
12. READ SMS: JSONRPC METHOD.....	54
13. Delete SMS: HTTP GET method.....	58
14. Delete SMS: JSONRPC method.....	59
15. Get outgoing queue length: HTTP GET method.....	60
16. Get outgoing queue length: JSONRPC method.....	61
17. Get inbox length: HTTP GET method.....	62
18. Get inbox length: JSONRPC method.....	63
19. Get sentitems length: HTTP GET method.....	64
20. Get sentitems length: JSONRPC method.....	65
21. Get GSM/3G signal strength: HTTP GET method.....	66
22. Get GSM/3G signal strength: JSONRPC method.....	67
23. Phonebook group create: HTTP GET method.....	68
24. Phonebook group create: JSONRPC method.....	69
25. Phonebook group read: HTTP GET method.....	70
26. Phonebook group read: JSONRPC method.....	71
27. Phonebook group update: HTTP GET method.....	72
28. Phonebook group update: JSONRPC method.....	74
29. Phonebook group delete: HTTP GET method.....	75
30. Phonebook group delete: JSONRPC method.....	76
31. Phonebook group add contact: HTTP GET method.....	77
32. Phonebook group add contact: JSONRPC method.....	78
33. Phonebook group remove contact: HTTP GET method.....	80
34. Phonebook group remove contact: JSONRPC method.....	81
35. Phonebook contact create: HTTP GET method.....	82

36. Phonebook contact create: JSONRPC method .....	83
37. Phonebook contact read: HTTP GET method .....	84
38. Phonebook contact read: JSONRPC method.....	86
39. Phonebook contact update: HTTP GET method .....	88
40. Phonebook contact update: JSONRPC method.....	89
41. Phonebook contact delete: HTTP GET method.....	90
42. Phonebook contact delete: JSONRPC method .....	91
43. Call with termination: HTTP GET method .....	92
44. Call with termination: JSONRPC method .....	94
Plugins and integration manuals for NMS & Auth systems .....	96
Extras.....	97
Connecting directly to SMSEagle database .....	97
Injecting short SMS using SQL .....	97
Injecting long SMS using SQL .....	98
Database cleaning scripts .....	100
SNMP agent .....	101
Setting up SNMP v3 access control.....	104
Failover (HA-cluster) feature .....	106
Forwarding logs to external server.....	109
Automatic software updates checks .....	110
Troubleshooting.....	112
Verification of LEDs.....	112
Checking the device logs.....	112
When the device is not reachable .....	112
Restoring factory defaults .....	113
Service & Repair.....	116
Warranty .....	116
Service .....	116
Tech Specs & Safety Information .....	118
Technical Specification .....	118

Important Safety Information .....	120
Regulatory Statements .....	121
FCC compliance statement .....	121
Canadian regulatory statement .....	121
Disposal and recycling information .....	122



GET READY  
TO START

## WHAT'S IN THE BOX

---

Your SMSEagle box contains:

- SMSEagle hardware SMS gateway
- 2x External omnidirectional 3dBi GSM/3G 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 3G antenna

---

Plug in both antenna connectors to the device. Make sure the antennas remain with 20cm spacing from each other.

### STEP 2: Insert SIM Card

---

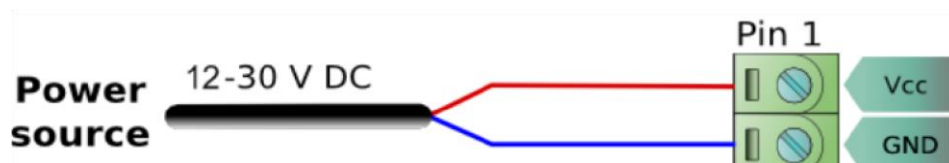
**Please install SIM Card when the device is SWITCHED OFF.** SIM Card slots is located at the bottom of the device. Use a ball-pen or small screwdriver to eject SIM Card tray. Insert card into tray and push it gently into slot.



### 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 / 1A. In order to power the device simply plug in a connector from AC/DC adaptor into the device.



### STEP 4: Configure IP settings

---



## PREPARE FOR FIRST START

---

### SMSEAGLE DEFAULT NETWORK CONFIGURATION:

#### **DHCP CLIENT IS ON**

(IP ADDRESS WILL BE OBTAINED AUTOMATICALLY FROM YOUR DHCP SERVER)

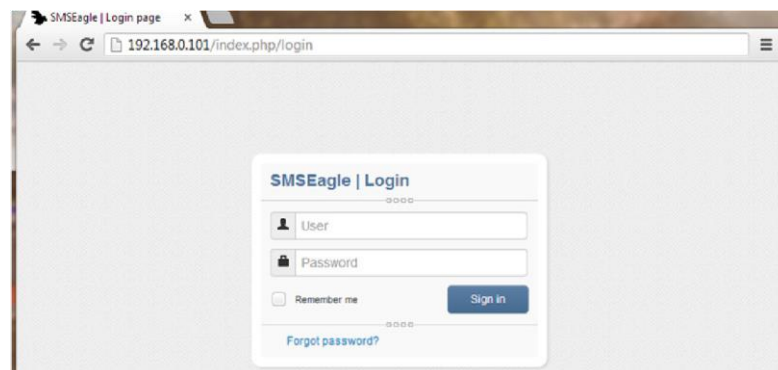
### A) CONNECT SMSEAGLE TO YOUR LAN AND **OBTAIN IP ADDRESS AUTOMATICALLY**

- connect the device to your LAN using Ethernet cable
- SMSEagle will obtain IP address automatically from your DHCP
- read assigned IP address on your DHCP server

### B) **OR SET IP ADDRESS FOR SMSEAGLE MANUALLY**

- connect a display using HDMI connector, connect a keyboard to USB port (note: cables are not provided)
- login to the SSH console using root credentials (these were provided with your device)
- edit configuration file with command:  
`mcedit /mnt/nand-user/smseagle/syscfg`  
change the following lines:  
HOST\_IP= *(set IP address for your device)*  
GW\_IP= *(default gateway IP address)*  
NET\_MASK= *(set subnet mask)*  
START\_DHCP=Y *(set to START\_DHCP=N to disable DHCP client)*
- save and exit the file
- shutdown the device
- now connect SMSEagle to your LAN using Ethernet cable

### C) **LOG IN TO SMSEAGLE**



Open an internet browser on your PC and go to the IP address assigned to your gateway

---

## PREPARE FOR FIRST START

---

**SMSEAGLE** DEFAULT USER IS:

**Username:** admin

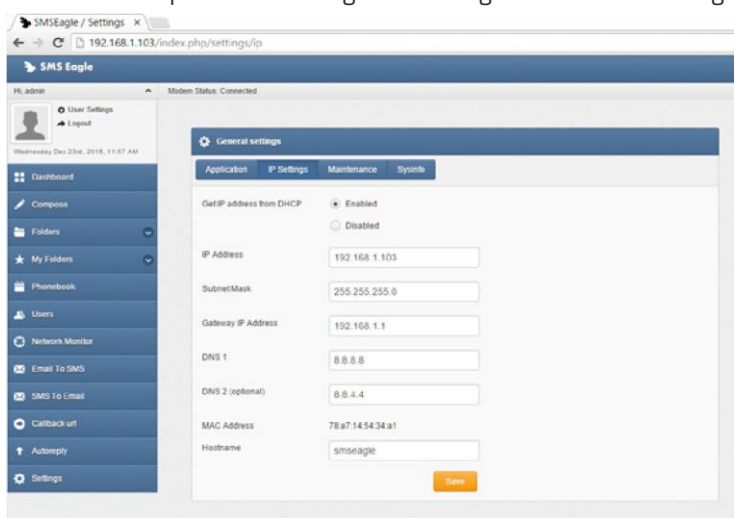
**Password:** password

*Login to application with above username and password.*

---

### D) CONFIGURE STATIC IP SETTINGS IN WEB-GUI (OPTIONAL)

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



Disable DHCP server. Enter your IP settings. > Press "Save" button.



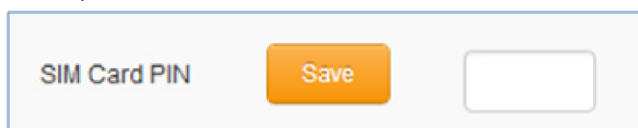
### STEP 5: Setting SIM Card PIN

---

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"



> Press "Save" button.

### STEP 6: Reboot the device

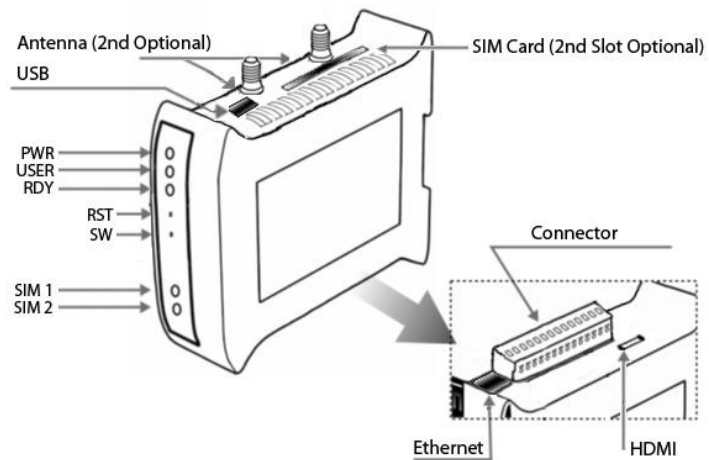
---

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



# USING OF SMSEAGLE

## GET TO KNOW WITH CONNECTORS, PORTS AND LEDs



Element	Label	Description
<b>Connector 1</b>	C1	Power connector and serial ports
<b>SIM Card Slot</b>	SIM1, SIM2 (optional)	SIM card slot(s)
<b>HDMI port</b>	HDMI	HDMI port (cable not included)
<b>USB port</b>	USB	USB port (cable not included)
<b>Ethernet Port</b>	ETH	Ethernet RJ45 socket
<b>Antenna</b>	ANT	Antenna socket
<b>Power LED</b>	PWR	LED indicating power-on
<b>User LED</b>	USER	LED for user application purpose (not used)
<b>SIM1,2 LEDs</b>	3G modem 1, 3G modem 2 (optional)	LED indicator for modem status
<b>Ready LED</b>	RDY	LED indication device status
<b>Reset</b>	RST	Switch for rebooting the device
<b>User Switch</b>	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 **Prepare for 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)

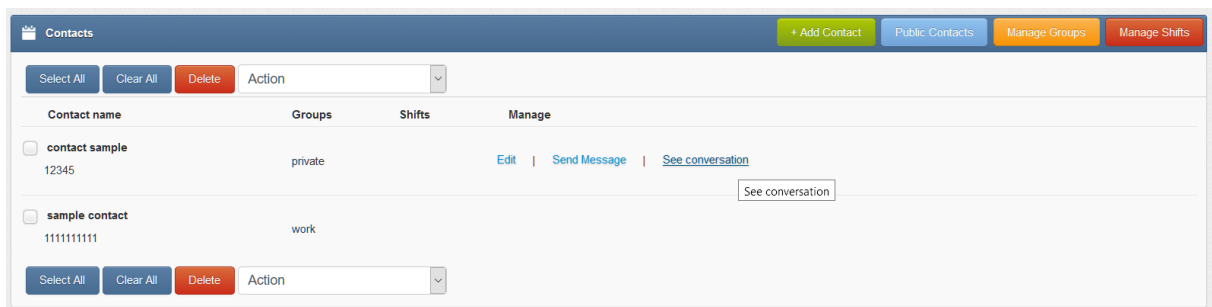
## Phonebook

---

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.



*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



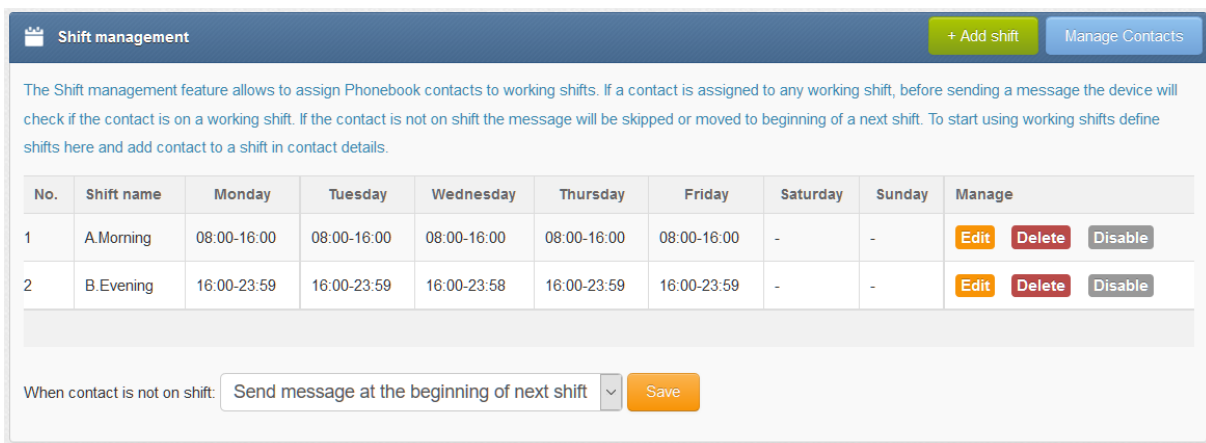
*Screenshot taken from phonebook groups*

In Phonebook Group Management view users can:

- Add/edit/delete groups
- Set groups to public or private visibility
- View group content (contacts belonging 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.



*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 PDF or CSV file.

The screenshot shows the 'Reporting module' interface with the 'Reports' tab selected. It includes search filters for date range (2017-05-30 to 2017-05-30, 00:00 to 10:59), folder (Inbox), sender number, and message content. A 'Choose output fields' dialog is open, showing 'SMS Center Number' and 'Read' as selectable items, and 'Receiving date', 'Sender number', 'Message ID', and 'Text' as selected items. A 'Generate report' button is visible. Below the filters, it states 'Number of messages matching your criteria: 1' and displays a table with one message.

Receiving date	Sender number	Message ID	Text
2017-05-30 10:32:11	Operator	912	Numer na ktory wyslales SMSa jest nieprawidlowy, sprawdz numer i sprobuj ponownie.

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.

The screenshot shows the 'Reporting module' interface with the 'Statistics' tab selected. It includes search filters for date range (2017-05-30 to 2017-05-30, 00:00 to 23:59), folder (Inbox), and checkboxes for 'Merge multipart messages' and 'Include messages from My Folders'. A 'Generate report' button is visible. Below the filters, it states 'Number of messages matching your criteria: 4' and displays a table with daily statistics.

Date	Sender number	Quantity
2017-05-30	Operator	4

Screenshot of Statistics view in Reporting module

## SMSEAGLE PLUGINS

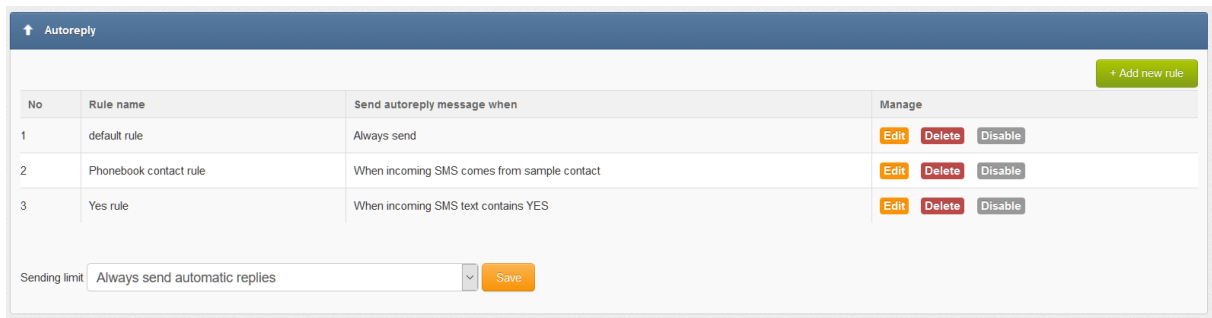
Basic features of SMSEagle software are extended by plugins that provide extra features to the software. Below you will find a description of plugins available in each SMSEagle device. All plugins are an integral part of SMSEagle software. That means that all described plugins are installed in a standard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

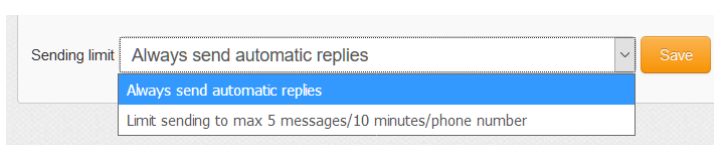


*Screenshot from plugin main window*

For each rule user can define:

- When autoreply message should be sent:
  - always,
  - when incoming message contains defined text,
  - when message sender belongs to Phonebook contact/group
- If autoreply message text should be sent as Unicode characters
- User may define many forwarding rules in the plugin
- Each rule is processed independently
- There is a possibility to enable/disable each rule

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



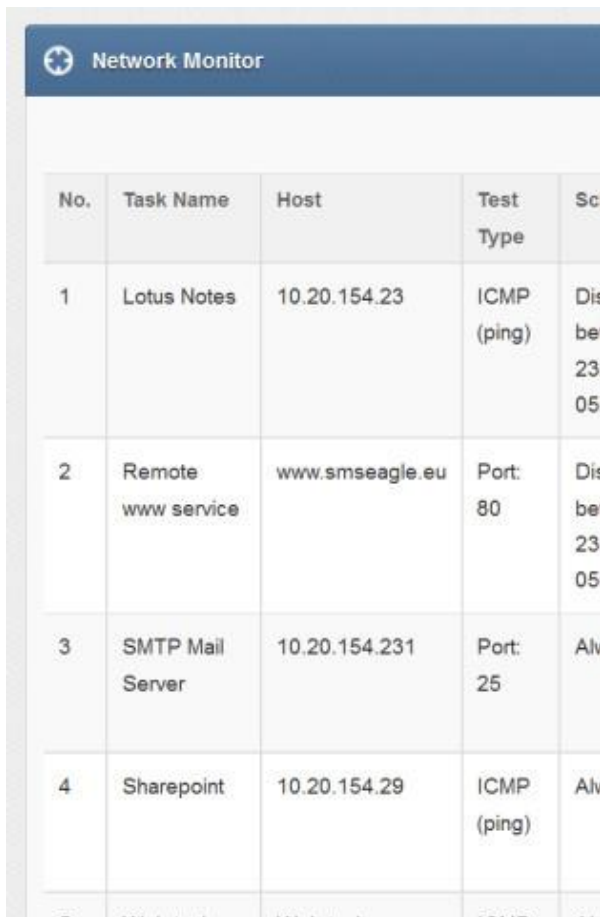
*Screenshot form "Add/edit autoreply rule"*

## Network Monitoring plugin

---

SMSEagle is equipped with network monitoring features. With that features you can monitor any device or service that has listening port open. SMSEagle Network Monitoring plugin sequentially controls availability of defined hosts/ports in Network Monitoring feature and sends defined SMS alert when port is unavailable/goes back to life. Below you will find a brief overview of plugin capabilities.

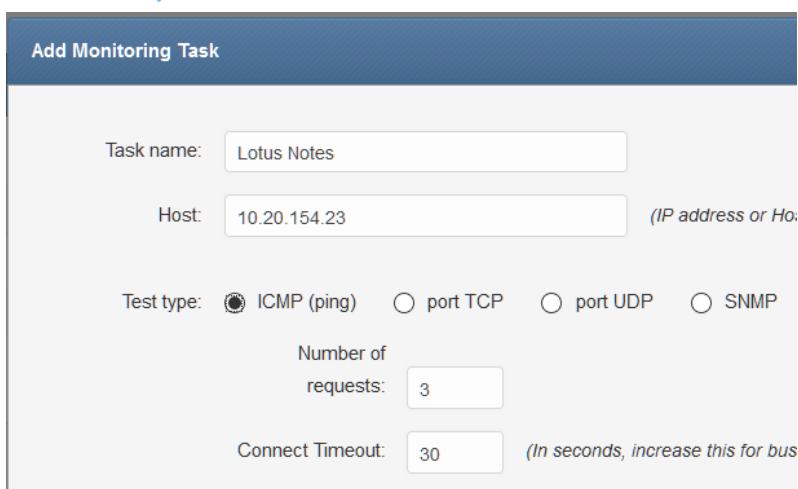
## Control status of all your defined tasks



No.	Task Name	Host	Test Type	St
1	Lotus Notes	10.20.154.23	ICMP (ping)	Dis bel 23: 05:
2	Remote www service	www.smseagle.eu	Port: 80	Dis bel 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:

Host:  (IP address or Ho

Test type:  ICMP (ping)  port TCP  port UDP  SNMP

Number of requests:

Connect Timeout:  (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

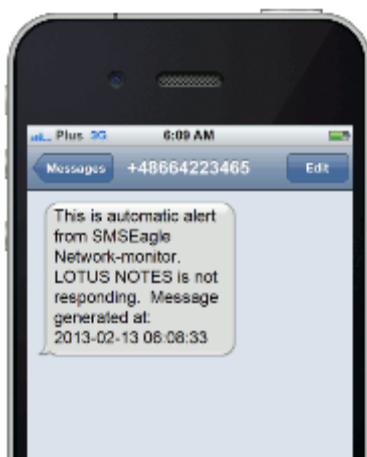
- choose if task should be always enabled...
- ...or disable it in chosen times  
(during a night, when a machine goes through planned restarts, during resource intensive backups, etc.)
- enter a phone number or choose a group of users to send your SMS alert to
- select when to send SMS alert (when host/service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

### Define a SMS alert message

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



- 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](http://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	<input type="text" value="Yes"/>
Email2SMS service status	Enabled
What to do with email subject	<input type="text" value="Include in SMS"/>
	<p>If authentication is enabled provide SMSEagle user and password in the email subject.</p> <p>Use the following syntax: login=john&amp;pass=doe (replace john doe with your own user and pass)</p>
Maximum number of characters in SMS	<input type="text" value="1300"/>
	Value should be between 1 and 1300
Unicode encoding of SMS text	<input type="text" value="No"/>
	This should be enabled only when you want to include special national

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 ääöä 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](mailto:smseagle@mycompany.com)

FROM: [john.doe@mycompany.com](mailto:john.doe@mycompany.com)

SUBJECT: +48333444555

BODY: Hello world!

In this case SMSEagle gateway will fetch incoming email from [smseagle@mycompany.com](mailto: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

Email2SMS poller service status **Enabled**

Check for email every   
Time in seconds

Maximum number of characters in SMS   
Value should be between 1 and 1300

Unicode encoding of SMS text   
This should be enabled only when you want to include special national chars (like 我) in SMS message

Protocol

Host

Port   
Standard email services ports: POP3: 110, POP3 (TLS/SSL): 995, IMAP: 143, IMAP (TLS/SSL): 993

Username

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

---

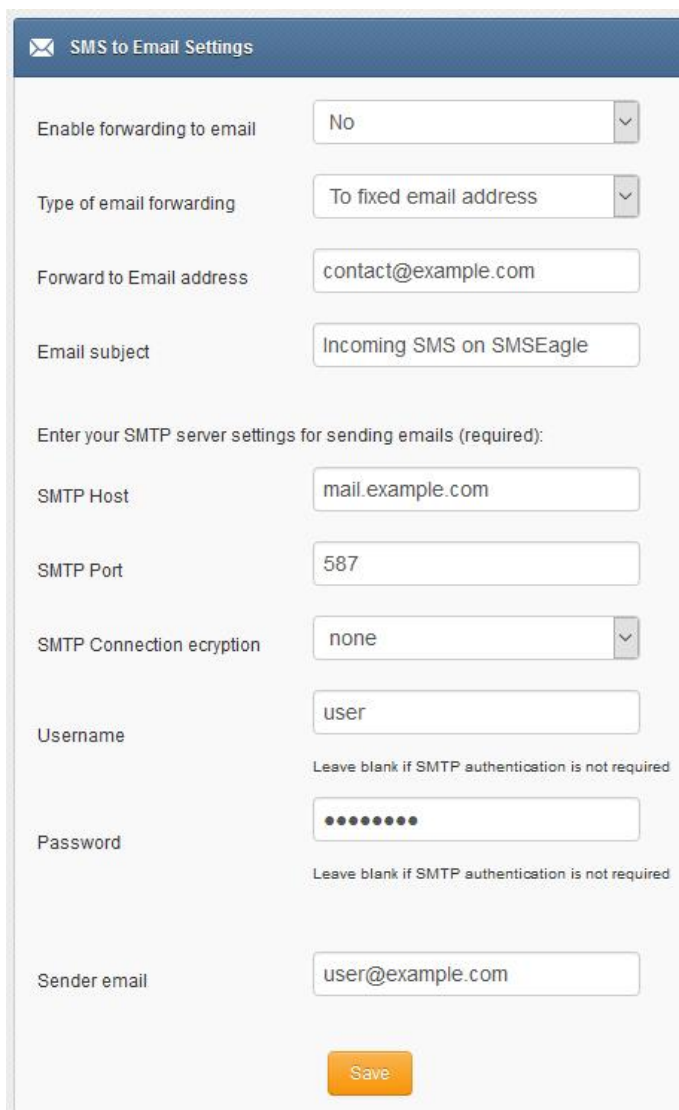
SMS to Email plugin allows you to forward incoming SMS messages to email address.

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 Email2SMS. 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.

### PLUGIN CONFIGURATION



The screenshot shows the 'SMS to Email Settings' configuration form. It includes the following fields and options:

- Enable forwarding to email:** A dropdown menu set to 'No'.
- Type of email forwarding:** A dropdown menu set to 'To fixed email address'.
- Forward to Email address:** A text input field containing 'contact@example.com'.
- Email subject:** A text input field containing 'Incoming SMS on SMSEagle'.
- Enter your SMTP server settings for sending emails (required):**
  - SMTP Host:** A text input field containing 'mail.example.com'.
  - SMTP Port:** A text input field containing '587'.
  - SMTP Connection encryption:** A dropdown menu set to 'none'.
  - Username:** A text input field containing 'user'.
  - Password:** A text input field with masked characters (dots). Below the field is the text 'Leave blank if SMTP authentication is not required'.
- Sender email:** A text input field containing 'user@example.com'.

At the bottom of the form is an orange 'Save' button.

### *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

### 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

#### **Sample email text:**

From: +483334455 (John Doe)

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

Message: My SMS message

### 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 request to a defined URL. HTTP request can be of type GET or POST.

## PLUGIN CONFIGURATION

### Callback url settings

Enable callback of custom url on incoming SMS

URL

URL method

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

Allow self-signed SSL certificate

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 \n  
**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 n  
**apikey:** API key of your service (optional)

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

**Request string example for HTTP GET:**  
`?sender=48601123123&timestamp=20140531092257&msgid=431&modemno=14`

Screenshot from Callback URL settings

- If you want to use the plugin, set 'Enable callback' to 'Yes'
- 'URL' field defines remote address of your callback script
- With 'URL method' you can choose whether callback to your URL will be done with HTTP GET or POST method
- Optionally you can define API key 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
- '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

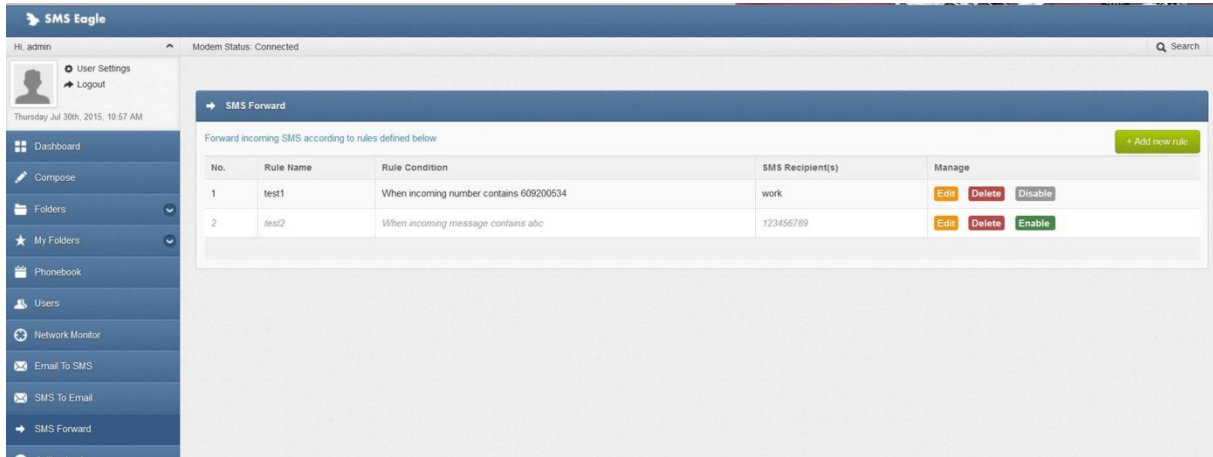
After sending HTTP GET/POST request to your callback URL, SMSEagle will be expecting HTTP response: 200 [OK]. If other or no response is received from your callback URL, SMSEagle will keep retrying every 2 minutes for 24 hours.

## SMS Forward

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

### PLUGIN CONFIGURATION

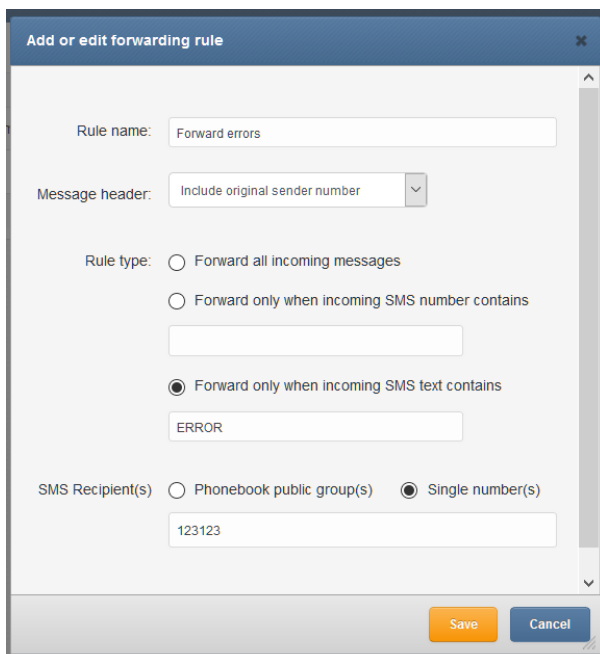
Plugin “SMS Forward” allows to add many forwarding rules. Each rule can be enabled or disabled by user.



*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.



*Screenshot form “Add/edit forwarding rule”*

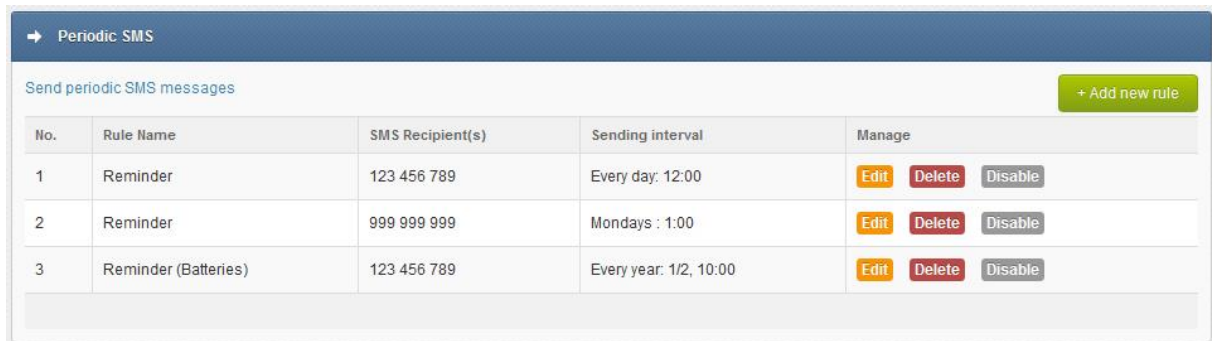
## Periodic SMS

---

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.



The screenshot shows the 'Periodic SMS' configuration window. At the top, there is a header 'Periodic SMS' with a right-pointing arrow. Below the header, the text 'Send periodic SMS messages' is displayed on the left, and a green button labeled '+ Add new rule' is on the right. The main content is a table with the following data:

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

*Screenshot from main plugin 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 (phone number(s) separated with comma or group(s) from phonebook)

Screenshot from "Add new rule" window

## Digital input/output

The NXS-family of SMSEagle devices is equipped with 2 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 logical states of inputs and outputs of SMSEagle NXS-family of devices are represented by the following voltages:

Logical level	Voltage
LOW (0)	0 V
HIGH (1)	+5V

### 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.



Plugin status: Enabled Save

### Digital inputs

Input 1 signal: 0 + Add new rule  
 Input 2 signal: 0

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

---

### Digital outputs

Output 1 signal: 0 + Add new rule  
 Output 2 signal: 0

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

Screenshot from plugin window

## DIGITAL INPUTS

For each processing rule for digital input user can define:

- The rule name
- Port number (1,2)
- State of input signal that will trigger sending of SMS message (field "When input signal")
- SMS text (field "Send SMS message")
- The recipient's name from phonebook

**Add or edit rule** ✕

Rule Name:

Port type:  ▼

Port number:  ▼

When input signal:  ▼

Send SMS message: 

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

Send to:

Insert Name from Contact List

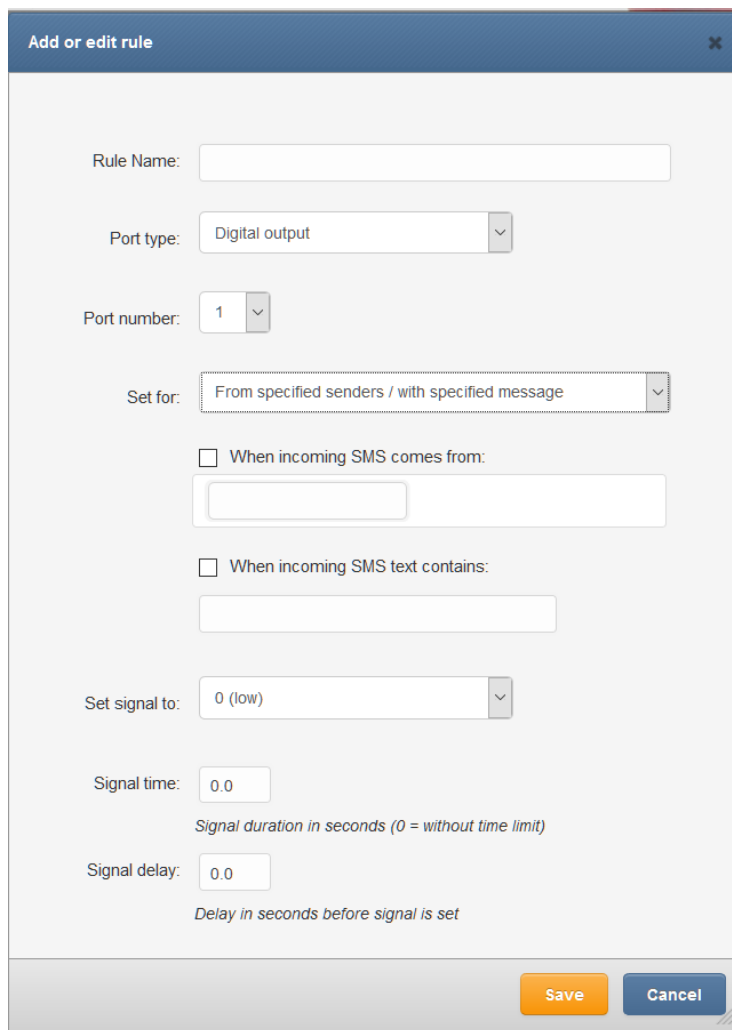
Save Cancel

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



The screenshot shows a window titled "Add or edit rule" with a close button (X) in the top right corner. The window contains the following fields and options:

- Rule Name:** A text input field.
- Port type:** A dropdown menu with "Digital output" selected.
- Port number:** A dropdown menu with "1" selected.
- Set for:** A dropdown menu with "From specified senders / with specified message" selected.
- When incoming SMS comes from:  
A text input field below the checkbox.
- When incoming SMS text contains:  
A text input field below the checkbox.
- Set signal to:** A dropdown menu with "0 (low)" selected.
- Signal time:** A text input field with "0.0" entered.  
Signal duration in seconds (0 = without time limit)
- Signal delay:** A text input field with "0.0" entered.  
Delay in seconds before signal is set

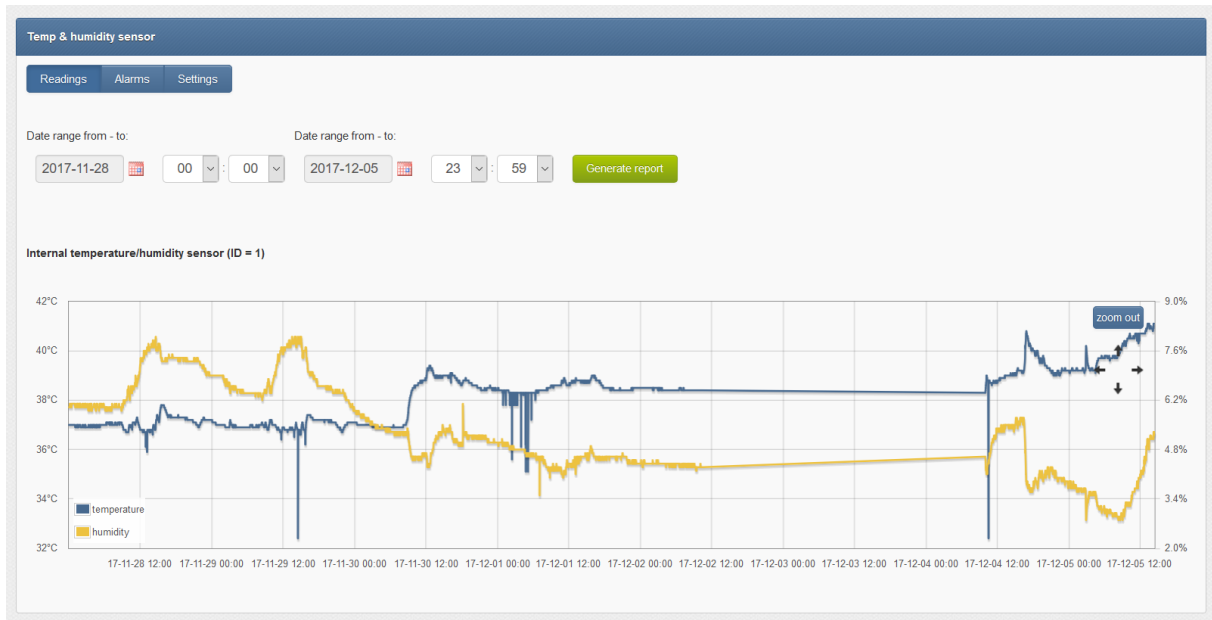
At the bottom right of the window, there are two buttons: "Save" (orange) and "Cancel" (blue).

Screenshot from digital output "Add or edit rule" window

**WARNING:** There is no over-voltage protection on the board. Randomly plugging wires and power sources into your digital inputs may kill it. Digital outputs has a current efficiency of 200mA. Consider using a relay for high power output devices.

## Temperature & humidity sensor

The NXS-family of SMSEagle devices is equipped with internal temperature and humidity sensor. The DHT22 sensor allows to measure temperature with  $\pm 0.5^{\circ}\text{C}$  accuracy and humidity with  $\pm 2\%$  RH accuracy. The measured temperature and humidity values can be displayed in web-gui of SMSEagle and used to trigger SMS message to single/many recipients.



Screenshot from plugin main window

### PLUGIN CONFIGURATION - ALARMS

Tab "Alarms" allows to define triggering rules for SMS alarms for temperature and humidity. User may define several processing rules.

The screenshot shows the 'Alarms' configuration window for the 'Temp & humidity sensor'. It features a table with the following data:

No.	Rule name	Alert when	Send to	Man
1	Humidity alarm	humidity is lower than 30.0 %	John Kowalski	Ed
2	Temperature alarm	temperature is higher than 60.0 °C	John Doe	Ed

Screenshot from "Alarms" window

For each processing rule for digital output user can define:

- The rule name
- Sensor (currently only 1 sensor is available)
- On what condition SMS alarm should be sent (temperature/humidity is higher/lower than given value)
- SMS text

- SMS recipient: contact name or group name from Phonebook

Rule name: Temperature alarm

Sensor: Internal temperature/humidity sensor (ID = 1)

Alert when: temperature is higher than 60.0 °C

SMS Message: Warning! Internal temperature on SMSEagle device located in server room A3 has reached {READVALUE}

Placeholders for SMS Text:  
{READVALUE} - value from sensor  
{TIMESTAMP} - read time

Send to: John Doe

Save Cancel

Screenshot from "Add or edit rule" window

## PLUGIN CONFIGURATION - SETTINGS

Tab "Settings" allows to control sensor settings. User may enable/disable sensor and set sensor reading period (in minutes).

Temp & humidity sensor

Readings Alarms Settings

Enable internal temperature/humidity sensor (ID = 1) Yes

Read sensor every 5  
Time in minutes

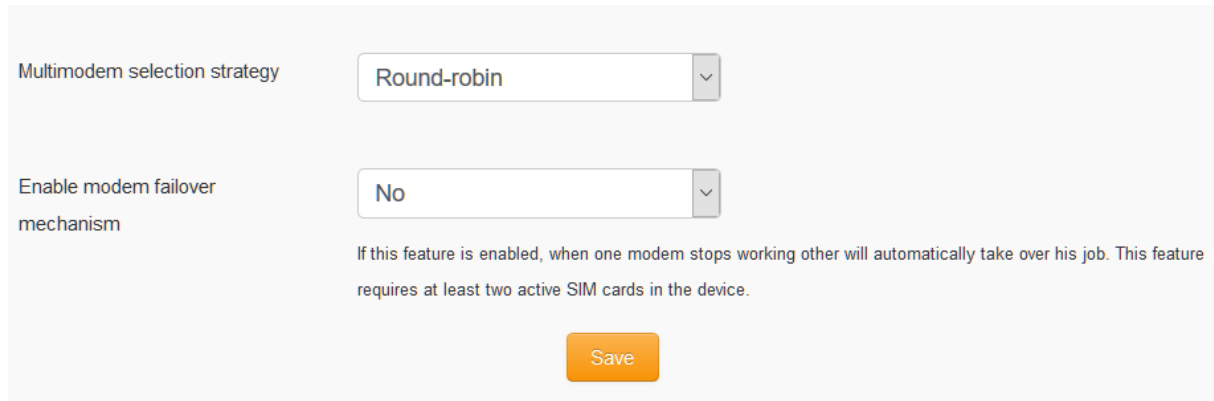
Save

Screenshot from "Settings" window

## MULTIMODEM FEATURES

---

SMSeagle NXS-9750 is equipped with two built-in modems. You can manage modem settings in web-gui menu Settings > Maintenance Tab. There are following options available for multimodem device:



Multimodem selection strategy

Enable modem failover mechanism

If this feature is enabled, when one modem stops working other will automatically take over his job. This feature requires at least two active SIM cards in the device.

### Multimodem selection strategy

This setting is responsible for modem choice strategy when sending SMS messages from SMSEagle. The following options are possible:

- Round-robin  
In this strategy modems are selected sequentially one-by-one when sending out SMS messages. This means that device sends messages using modem1 > modem2 > modem1 > modem2, etc.
- SIM1 as Master modem  
In this strategy modem1 is always selected when sending out SMS messages. If failover is enabled (see below) modem2 will be always used as a backup in failover strategy
- SIM2 as Master modem  
In this strategy modem1 is always selected when sending out SMS messages. If failover is enabled (see below) modem2 will be always used as a backup in failover strategy

### Enable modem failover mechanism

If this setting is enabled, when one modem stops working other will automatically take over his job. This feature requires at least two active SIM cards in the device. The health check for each modem is performed with 5 minutes frequency. If during a health check a modem is not connected to network the other will automatically take over his jobs (including messages waiting in Outbox folder).

## 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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	<i>(optional parameter)</i> 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	<i>(optional parameter)</i> 0 = normal SMS (default), 1 = SMS will be sent as flash message
oid	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> sending modem number (only for multimodem devices)

responsetype	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	<i>(optional parameter)</i> 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	<i>(optional parameter)</i> 0 = normal SMS (default), 1 = SMS will be sent as flash message
oid	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> sending modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> 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](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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem

unicode	<i>(optional parameter)</i> 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	<i>(optional parameter)</i> 0 = normal SMS (default), 1 = SMS will be sent as flash message
modem_no	<i>(optional parameter)</i> sending modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> 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=201401152132
```

```
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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	<i>(optional parameter)</i> 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	<i>(optional parameter)</i> 0 = normal SMS (default), 1 = SMS will be sent as flash message
modem_no	<i>(optional parameter)</i> sending modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> 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": "mymes sage"}}
or
{"method": "sms.send_togroup",
"params": {"login": "john", "pass": "doe", "groupname": "admins", "message": "mymes sage", "date": "201401152132"}}
or
{"method": "sms.send_togroup",
"params": {"login": "john", "pass": "doe", "groupname": "admins", "message": "mymes sage", "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](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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	<i>(optional parameter)</i> 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	<i>(optional parameter)</i> 0 = normal SMS (default), 1 = SMS will be sent as flash message
modem_no	<i>(optional parameter)</i> sending modem number (only for multimodem devices)

responsetype	<i>(optional parameter)</i> 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=201401152132
```

```
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>
  <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 contact doesn't exist):

```
<xml>
  <error_text>Invalid contact name – [contact_name]</error_text>
  <status>error</status>
</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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = normal priority, 1 = SMS will have higher priority in Outbox queue when processed by GSM-modem
unicode	<i>(optional parameter)</i> 0 = no Unicode encoding (default), 1 = SMS will be encoded using Unicode (you can send national characters)
flash	<i>(optional parameter)</i> 0 = normal SMS (default), 1 = SMS will be sent as flash message
modem_no	<i>(optional parameter)</i> sending modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> 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": "my message"}}
or
{"method": "sms.send_tocontact",
"params": {"login": "john", "pass": "doe", "contactname": "johndoe", "message": "my message", "date": "201401152132"}}
or
{"method": "sms.send_tocontact",
"params": {"login": "john", "pass": "doe", "contactname": "johndoe", "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 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](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	<i>(optional parameter)</i> sending modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> text = format response as text (default), xml = format response as XML object

### EXAMPLES:

[https://url-of-smseagle/index.php/http\\_api/send\\_ussd?](https://url-of-smseagle/index.php/http_api/send_ussd?)

login=john&pass=doe&to=\*101%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	<i>(optional parameter)</i> sending modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> 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](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	<i>(optional parameter)</i> UDH header for the message (in hex format)
data	binary message (in hex format)
class	<i>(optional parameter)</i> message class
modem_no	<i>(optional parameter)</i> sending modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> 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=EA0601AE02056A0045C6
0C03777772E736D736561676C652E657500080103534D534561676C65000101
```

#### 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	<i>(optional parameter)</i> UDH header for the message (in hex format)
data	binary message (in hex format)
class	<i>(optional parameter)</i> message class
modem_no	<i>(optional parameter)</i> sending modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> 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": "0605040B8423F0",
"data": "EA0601AE02056A0045C60C03777772E736D736561676C652E657500080103534D534561676C65000101"}}}
```

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](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	<i>(optional parameter)</i> minimal message-id
from	<i>(optional parameter)</i> telephone number of SMS sender (for inbox)
to	<i>(optional parameter)</i> telephone number of SMS receiver (for sentitems)
datefrom	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> how many messages to show
unread	<i>(optional parameter)</i> 1 = show only unread messages
responsetype	<i>(optional parameter)</i> 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 and higher. Limit number of messages to 5:

```
https://url-of-smseagle/index.php/http_api/read_sms?  
login=john&pass=doe&folder=sentitems&idfrom=1234&limit=5
```

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&date  
to=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>2016-11-17 13:23:47</UpdatedInDB>  
<ReceivingDateTime>2015-01-20 16:13:57</ReceivingDateTime>  
<Text>005400650073007400200032</Text>  
<SenderNumber>1234567</SenderNumber>  
<Coding>Default_No_Compression</Coding>  
<UDH/>  
<SMSCNumber>+48790998250</SMSCNumber>  
<Class>-1</Class>  
<TextDecoded>Test 2</TextDecoded>  
<ID>14</ID>  
<RecipientID/>  
<Processed>t</Processed>  
<id_folder>1</id_folder>  
<readed>>true</readed>  
<last_reply/>  
<oid/>  
</item>  
<item>  
<UpdatedInDB>2016-11-17 13:36:10</UpdatedInDB>
```

```

<ReceivingDateTime>2016-06-16 14:47:10</ReceivingDateTime>
<Text>004F00640070002000320020</Text>
<SenderNumber>1234</SenderNumber>
<Coding>8bit</Coding>
<UDH/>
<SMSCNumber>+48790998250</SMSCNumber>
<Class>-1</Class>
<TextDecoded>8b123sad</TextDecoded>
<ID>24</ID>
<RecipientID>smseagle1</RecipientID>
<Processed>t</Processed>
<id_folder>1</id_folder>
<readed>>true</readed>
<last_reply>Admin</last_reply>
<oid/>
</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>

```

## 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	<i>(optional parameter)</i> minimal message-id
from	<i>(optional parameter)</i> telephone number of SMS sender (for inbox)
to	<i>(optional parameter)</i> telephone number of SMS receiver (for sentitems)
datefrom	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> how many messages to show
unread	<i>(optional parameter)</i> 1 = show only unread messages
responsetype	<i>(optional parameter)</i> 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 and higher. Limit number of messages to 5:

```
{ "method": "sms.read_sms",
  "params": { "login": "john", "pass": "doe", "folder": "sentitems", "idfrom": "1234", "limit": "5" } }
```

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": "7654321", "datefrom": "20141224081000", "dateto": "20141231235959" } }
```

#### RESPONSE:

Sample response (inbox folder):

```
{
  "result": [
```

```

{
  "UpdatedInDB": "2016-11-14 10:15:58",
  "ReceivingDateTime": "2015-01-20 16:13:57",
  "Text": "005400650073007400200032",
  "SenderNumber": "+48123456789",
  "Coding": "Default_No_Compression",
  "UDH": "",
  "SMSCNumber": "+48790998250",
  "Class": "-1",
  "TextDecoded": "Test 2",
  "ID": "14",
  "RecipientID": "",
  "Processed": "t",
  "id_folder": "1",
  "readed": "true",
  "last_reply": null,
  "oid": null
},
{
  "UpdatedInDB": "2016-11-14 10:15:58",
  "ReceivingDateTime": "2016-06-16 14:27:10",
  "Text": "004F0064007000200031",
  "SenderNumber": "+48987654321",
  "Coding": "Default_No_Compression",
  "UDH": "",
  "SMSCNumber": "+48790998250",
  "Class": "-1",
  "TextDecoded": "Odp 1",
  "ID": "23",
  "RecipientID": "smseagle1",
  "Processed": "t",
  "id_folder": "1",
  "readed": "true",
  "last_reply": "Tester",
  "oid": "234234"
},
{
  "UpdatedInDB": "2016-11-14 10:15:58",
  "ReceivingDateTime": "2016-06-16 14:47:10",
  "Text": "004F00640070002000320020",
  "SenderNumber": "1234",
  "Coding": "8bit",
  "UDH": "",
  "SMSCNumber": "+48790998250",
  "Class": "-1",
  "TextDecoded": "8b123sad",
  "ID": "24",
  "RecipientID": "smseagle1",
  "Processed": "t",
  "id_folder": "1",
  "readed": "true",
  "last_reply": "Tester",
  "oid": ""
}
]

```



```
}
```

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 (sentitems folder):

```
{
  "result": {
    "messages": [
      {
        "UpdatedInDB": "2014-12-05 11:58:03",
        "InsertIntoDB": "2014-12-05 10:57:38",
        "SendingDateTime": "2016-06-16 14:27:49",
        "DeliveryDateTime": null,
        "Text": "0074006500730074",
        "DestinationNumber": "123456789",
        "Coding": "Default_No_Compression",
        "UDH": "",
        "SMSCNumber": "+48602951111",
        "Class": "1",
        "TextDecoded": "test",
        "ID": "61",
        "SenderID": "smseagle1",
        "SequencePosition": "1",
        "Status": "SendingError",
        "StatusError": "-1",
        "TPMR": "-1",
        "RelativeValidity": "255",
        "CreatorID": "admin",
        "id_folder": "3"},
      {
        "UpdatedInDB": "2016-06-16 14:48:56",
        "InsertIntoDB": "2016-06-16 12:48:45",
        "SendingDateTime": "2016-06-16 14:48:56",
        "DeliveryDateTime": null,
        "Text": "0074006500730074006F007500740033",
        "Coding": "Default_No_Compression",
        "UDH": "",
        "SMSCNumber": "+48602951111",
        "Class": "-1",
        "TextDecoded": "testout3",
        "ID": "384",
        "SenderID": "smseagle1",
        "SequencePosition": "1",
        "Status": "SendingOKNoReport",
        "StatusError": "-1",
        "TPMR": "250",
        "RelativeValidity": "255",
        "CreatorID": "admin",
```

```

        "id_folder": "3"
    }
],
    "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"}}

```

### 13. Delete SMS: HTTP GET method

#### HTTP GET METHOD:

[https://url-of-smseagle/index.php/http\\_api/delete\\_sms](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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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","idto":"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](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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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](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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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](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	<i>(optional parameter)</i> modem number to be queried (default = 1). Used only in multimodem devices
responsetype	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> modem number to be queried (default = 1). Used only in multimodem devices
responsetype	<i>(optional parameter)</i> simple = format response as simple object with one result field (default), extended = format response as extended JSON object

### BODY:

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

### RESPONSE:

Response: {"result": GSM 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](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	<i>(optional parameter)</i> 0 = private group, 1 = public group
responsetype	<i>(optional parameter)</i> 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](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	<i>(optional parameter)</i> 0 = private group, 1 = public group
responsetype	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = private group (default value), 1 = public group
uid	<i>(optional parameter)</i> id of user who created the group
responsetype	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = private group (default value), 1 = public group

uid	<i>(optional parameter)</i> id of user who created the group
responsetype	<i>(optional parameter)</i> 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](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	<i>(optional parameter)</i> 0 = private group, 1 = public group
responsetype	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = private group, 1 = public group
responsetype	<i>(optional parameter)</i> 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": "myusers",
"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](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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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": "myusers" }
}
```

#### 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](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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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](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	<i>(optional parameter)</i> 0 = private contact, 1 = public contact (default value)
responsetype	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = private contact 1 = public contact (default value)
responsetype	<i>(optional parameter)</i> 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](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	<i>(optional parameter)</i> 0 = private contacts (default value), 1 = public contacts

uid	<i>(optional parameter)</i> id of user who created the contact
responsetype	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = private contacts (default value), 1 = public contacts
uid	<i>(optional parameter)</i> id of user who created the contact
responsetype	<i>(optional parameter)</i> 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": "false"  
  }  
] }
```

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": "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 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	<i>(optional parameter)</i> 0 = private group, 1 = public group
responsetype	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 0 = private group, 1 = public group
responsetype	<i>(optional parameter)</i> 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": "john1ord", "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](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	<i>(optional parameter)</i> 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	<i>(optional parameter)</i> 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": "john1ord" } }
```

#### 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. Call with termination: HTTP GET method

#### HTTP GET METHOD:

[https://url-of-smseagle/index.php/http\\_api/call\\_with\\_termination](https://url-of-smseagle/index.php/http_api/call_with_termination)

Parameter	Description
login	your user to login to SMSEagle
pass	your password to login to SMSEagle
to	phone number to call
duration	connection duration (in seconds)
modem_no	<i>(optional parameter)</i> calling modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> text = format response as text (default), xml = format response as XML object

#### EXAMPLES:

`https://url-of-smseagle/index.php/http_api/call_with_termination?login=john&pass=doe&to=12345&contactname=johnlord`

#### RESPONSE:

Response: **OK**

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

Response (when modem doesn't support voice calls): **This modem doesn't support voice calls**

Response (when wrong parameters):

**Wrong or missing >>to<< parameter**

**Wrong or missing >>duration<< parameter**

Response (when modem\_no is wrong): **Modem not recognized**

#### 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 >>duration<< parameter</error_text>
  <status>error</status>
</xml>
```

Response (when modem doesn't support voice calls):

```
<xml>
```

```

    <error_text>This modem doesn't support voice calls</error_text>
    <status>error</status>
</xml>

```

Response (when modem\_no is wrong):

```

<xml>
  <error_text> Modem not recognized </error_text>
  <status>error</status>
</xml>

```

#### 44. Call with termination: 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
to	phone numer to call
duration	connection duration (in seconds)
modem_no	<i>(optional parameter)</i> calling modem number (only for multimodem devices)
responsetype	<i>(optional parameter)</i> text = format response as text (default), xml = format response as XML object

##### EXAMPLES:

```

{"method":"phone.call_with_termination",
"params":{"login":"john","pass":"doe","to":"12345", "duration":"5"}}

```

##### RESPONSE:

Response: {"result": "OK"}

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

Response (when modem doesn't support voice calls): {"result": "This modem doesn't support voice calls"}

Response (when wrong parameters):

```

{"result": "Wrong or missing >>to<< parameter"}
{"result": "Wrong or missing >>duration<< parameter"}

```

Response (when modem\_no is wrong): {"result": "Modem not recognized"}

##### RESPONSE (EXTENDED):

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

Response (when wrong logindata):

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

Response (when modem doesn't support voice calls):

```
{"result": {"error_text": "This modem doesn't support voice calls", "status": "error"}}
```

Response (when wrong parameters):

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

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

Response (when modem\_no is wrong):

```
{"result": {"error_text": "Modem not recognized", "status": "error"}}
```

## 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/>



## EXTRAS

---

### 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:

#### POSTGRES SQL DATABASE CREDENTIALS

Host: IP address of your device

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". Field SenderID contains identification number of SMSEagle modem. For modem 1 SenderID = smseagle1 and for modem 2 SenderID = smseagle2.

## [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.*

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

1) Create a file inside `/etc/cron.d/` directory with your desired name (eg. *pico db\_cleaner*)

2) Example content of this file:

```
0 0 1 * * root /mnt/nand-user/scripts/db_delete_allfolders
```

This will run cleaning script every 1<sup>st</sup> day of month.

“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
<b>GSM_Signal</b>	Returns GSM/3G signal strength in percent. Value range: 0-100. If modem is disconnected from GSM/3G network GSM_Signal returns 0.	.1.3.6.1.4.1.8072.1.3.2.3.1.2.11.71 .83.77.95.83.105.103.110.97.108. 49
<b>GSM_NetName1</b>	Returns Network Name used on current SIMcard at slot #1	.1.3.6.1.4.1.8072.1.3.2.3.1.2.12.71 .83.77.95.78.101.116.78.97.109.1 01.49
<b>GSM_NetName2</b>	Returns Network Name used on current SIMcard at slot #2	.1.3.6.1.4.1.8072.1.3.2.3.1.2.12.71 .83.77.95.78.101.116.78.97.109.1 01.50
<b>FolderOutbox_Total</b>	Returns number of SMS messages in Outbox folder (outgoing queue length)	.1.3.6.1.4.1.8072.1.3.2.3.1.2.18.70 .111.108.100.101.114.79.117.11 6.98.111.120.95.84.111.116.97.1 08
<b>FolderInbox_Total</b>	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
<b>FolderSent_Last24H</b>	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 .111.108.100.101.114.83.101.11 0.116.95.76.97.115.116.50.52.72
<b>FolderSent_Last1M</b>	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 .111.108.100.101.114.83.101.11 0.116.95.76.97.115.116.49.77
<b>FolderSent_Last24HSendErr</b>	Returns number of SMS messages sent with error within last 24h. Error occurs when 3G modem cannot send SMS message or message is rejected by GSM	.1.3.6.1.4.1.8072.1.3.2.3.1.2.25.70 .111.108.100.101.114.83.101.11 0.116.95.76.97.115.116.50.52.72. 83.101.110.100.69.114.114

	/3G carrier (mostly happens when a credit on pre-paid SIM card is over)	
--	---	--

## 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  
.1.3.6.1.4.1.8072.1.3.2.3.1.2.11.71.83.77.95.83.105.103.110.97.108.49
```

Result:

```
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."GSM_Signal" = STRING: 54
```

*Comment: GSM/3G Signal strength value is 54%*

### EXAMPLE OF READING **GSM\_NETNAME1** VALUE USING NET-SNMP LIBRARY

a) Command for reading the result value:

```
snmpget -v 2c -c public localhost  
.1.3.6.1.4.1.8072.1.3.2.3.1.2.12.71.83.77.95.78.101.116.78.97.109.101.49
```

Result:

```
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."GSM_NetName1" = STRING: PLAY
```

*Comment: Currently used network at SIM card #1 is PLAY*

#### 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_NetName1" = STRING: PLAY
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."GSM_NetName2" = STRING: PLAY
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

The screenshot shows the iReasoning MIB Browser interface. The address is 192.168.1.106 and the OID is .1.3.6.1.4.1.8072.1.3.2.3.1.2. The MIB tree on the left shows the path: iso.org.dod.internet > mgmt > private > enterprises > netSnmp > netSnmpObjects > nsExtensions > nsSnmpExtendMIB > nsExtendObjects > nsExtendNumEntries > nsExtendOutput1Table > nsExtendOutput1Entry > nsExtendOutput1Line > nsExtendOutputFull. The result table on the right shows the following data:

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	sentIm
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

The bottom panel shows the details for nsExtendOutputFull:

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`

## Failover (HA-cluster) feature

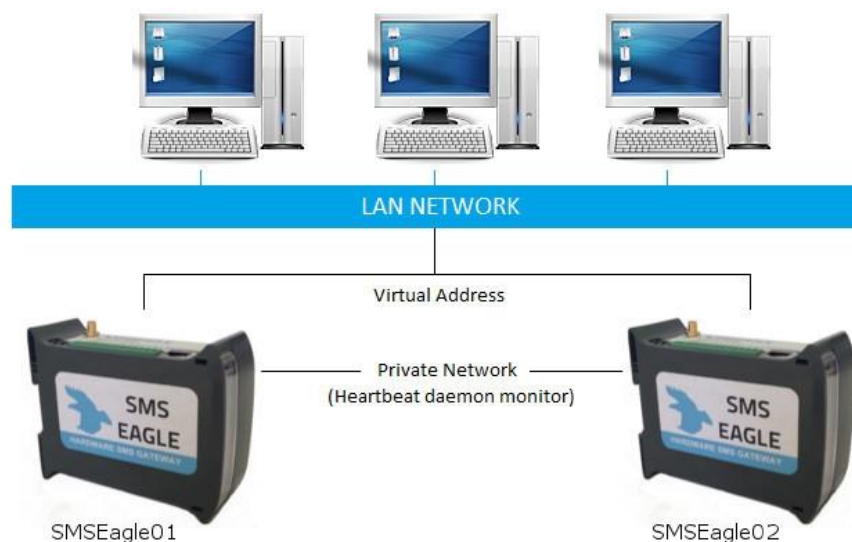
'High-availability clusters (also known as HA clusters or fail over clusters) are groups of computers (...) that can be reliably utilized with a minimum of down-time. They operate by using high availability software to harness redundant computers in groups or clusters that provide continued service when system components fail. Without clustering, if a server running a particular application crashes, the application will be unavailable until the crashed server is fixed. HA clustering remedies this situation by detecting hardware/software faults, and immediately restarting the application on another system or whole node without requiring administrative intervention, a process known as failover.' (source: Wikipedia)

SMSEagle NXS family has its own HA-cluster feature. For service to work you need at least 2 gateways ('aka' nodes). Our service monitors devices working in cluster and detects faults with following functions:

1. Apache2 WWW server
2. PostgreSQL database
3. SNMP agent
4. Modem software (Gammu-SMSD daemon)
5. Accessibility (response to ping) of whole node.

Every node in a cluster can have one of three states:

- Master: first healthy node which became available in a cluster, and is accessible through Virtual IP
- Backup: second healthy node in a cluster, ready and waiting for replacing Master when needed
- Fault: node with detected service fault



Basically devices (nodes) should see each other on the network. By default HA-nodes use 224.0.0.18 IP address for VRRP (Virtual Router Redundancy Protocol) for communication between two nodes for health check. If nodes are on the same network (same subnet & IP range) there is no need for any network configuration. If two nodes are behind firewalls, make sure firewall is configured to accept multicast and VRRP protocol (IP Protocol #112).

When the daemon running at MASTER device detects failure of at least one described features it immediately automatically switches cluster's IP assignment to the BACKUP device (node) providing continuous usage of the SMSEagle HA-cluster for the user.

#### HOW TO CONFIGURE FAILOVER (HA-CLUSTER):

Failover cluster can be easily configured using web-gui. Configuration can be done in menu "Settings" > tab "Failover". For each device in failover cluster:

- enter virtual IP address in the field "Virtual IP Address"
- set "Enable Failover cluster" to "Yes".

**Save** configuration. **Reboot** each device after saving.

General settings

Application IP Settings Failover Maintenance Sysinfo

Enable Failover cluster Yes

Failover status Disabled

Virtual IP Address 192.168.0.100

Please note:

- Failover (HA) cluster requires minimum 2 devices for operation
- Virtual IP address must be in the same subnet mask as the device's physical IP address
- First device to boot becomes MASTER in failover cluster, second and next are BACKUPS
- Result of a proper work of a failover cluster is one MASTER device, and at least one BACKUP device

Save

*Screenshot from "General settings-Failover"*

A device that boots first becomes MASTER in failover cluster, second and next devices are BACKUPS. In the cluster you have one MASTER device and at least one BACKUP device (depending of number of nodes). The HA-cluster will automatically switch between physical devices depending on the node health-check.

After correct configuration of the HA-cluster **you should access the cluster via its Virtual IP address.**

*Alternatively configuration may be also made using Linux command line:*

9. Log in via SSH using *root* account
10. Navigate to:  
`cd /mnt/nand-user/smseagle/keepalived/`
11. Configuration script:  
`./failover`

```
SMSEagle Failover Cluster feature presents:
Usage /mnt/nand-user/smseagle/keepalived/failover [start] | [stop] | [status] |
[config VIRTUAL_IP_ADDRESS]
  where:
    - start: add failover cluster to autorun and start the service
    - stop: remove failover cluster from autorun and stops the service
    - status: check the status of failover clusters node
    - config: set up Virtual IP Address
```

12. First run:

```
./failover config VIRTUAL_IP_ADDRESS
```

This will setup your Virtual IP address for your cluster

13. Then run:

```
./failover start
```

which will run the Failover service

14. You can check device state with:

```
./failover status
```

15. You can always disable service with:

```
./failover stop
```

***Above commands have to be done on each device (node) of HA-cluster!***

## SNMP-monitoring

Failover feature uses KEEPALIVED-MIB for SNMP monitoring.

### EXAMPLE OF READING DEVICE CLUSTER STATE VALUE USING NET-SNMP LIBRARY

a) Command for reading the result value:

```
snmpget -v 2c -c public ip-of-smseagle .1.3.6.1.4.1.9586.100.5.2.3.1.4.1
```

Result:

```
KEEPALIVED-MIB::vrrpInstanceState.1 = INTEGER: master(2)
```

*Comment: Current device state is master*

## 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](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:

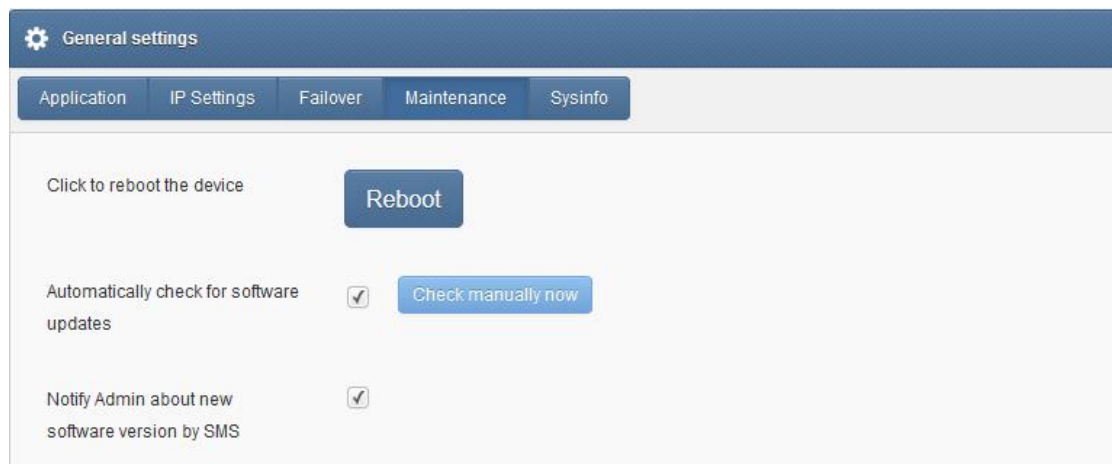
### MANUAL CHECK

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.



*Screenshot from "General settings-Maintenance"*

*Notice: Your SMSEagle device must have a HTTPS connectivity with address [www.smseagle.eu](https://www.smseagle.eu) in order for this feature to work.*



# TROUBLESHOOTING

## TROUBLESHOOTING

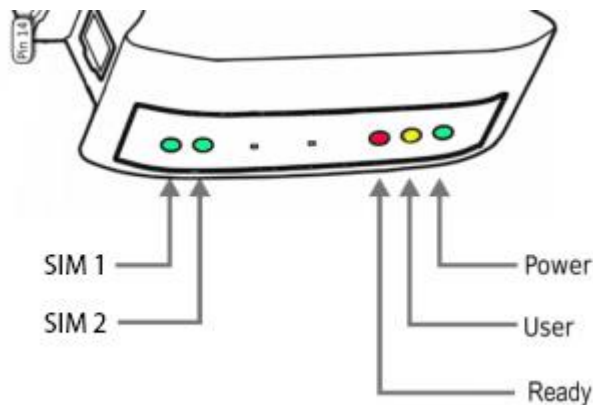
---

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
<b>Power (PWR)</b>	Continuously lit
<b>User</b>	Blinks during flashdisk read/write
<b>Ready (RDY)</b>	Blinking
<b>SIM1</b>	Slow flashing in stand-by mode, Quick flashing when modem 1 in use
<b>SIM2</b>	Slow flashing in stand-by mode, Quick flashing when modem 2 in use

### Checking the device logs

SMSEagle operates on Linux system. Linux system log is available under menu position "Settings" > "Sysinfo". 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.



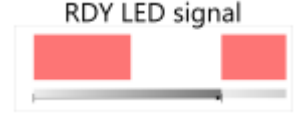



2. In the case when the device does not respond due to a malfunction or incorrect user settings please reboot the device by disconnecting and connecting power source (or pressing Reset switch).
3. If you still cannot connect with the device, it is possible to restore to factory IP settings by using the SW button.

### Restoring factory defaults

This action restores the following settings to default values: **IP settings, time zone settings, database content, Linux OS users/passwords**

In order to restore factory defaults proceed with the following steps:

LED signaling	USER actions	System reaction
<p>RDY LED signal</p> 	<p>1. <b>When the device is ready to operate</b></p>	
	<p>2. <b>Press and hold SW button for 10 seconds</b></p>	<p>Restore service is counting down.</p>
<p>USER LED signal</p> 	<p>3. <b>Release SW button after 10 seconds.</b> User LED will begin to blink.</p>	<p>System is reading factory defaults.  Factory settings are being applied to the device.</p>
<p>RDY LED signal</p>  <p>USER LED signal</p> 	<p>4. <b>Wait until system reboots.</b>  Default settings are restored.</p>	<p>System is going for a reboot.</p>

*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.*

# IV

## SERVICE & REPAIR

## SERVICE & 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](http://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](mailto: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 > Sysinfo. 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.

**V**

**TECH SPECS  
& SAFETY  
INFORMATION**

## Technical Specification

---

### HARDWARE SPECIFICATION

- Processor type:
  - Device model Rev.1: Broadcom BCM2835 0.7GHz ARM11
  - Device model Rev. 2: Broadcom BCM2837 1.2 GHz quad-core ARM Cortex-A53 (64-bit)
- Operational memory (RAM):
  - Device model Rev.1: 512 MB SDRAM @ 400 MHz
  - Device model Rev.2: 1GB LPDDR2 @ 900 MHz
- Network interface: Ethernet 10/100 TX (1xRJ45)
- 4GB Flash disk
- 1x USB 2.0 port
- 1x HDMI port
- 2x RS232 serial ports
- 2x DO/DI GPIO ports
- RTC Clock: RTC 240B SRAM, Watchdog timer
- Humidity & temperature sensor
- Power consumption: max 27W
- Noise level: Fan-less
- 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)
- 2x 3G Modem:
  - Waveband: GSM, UMTS
  - GSM/GPRS quad-band 850/900/1800/1900 MHz
  - UMTS 800/850/900/AWS 1700/1900/2100 MHz
- SIM card standard: mini

- Antenna connector: SMA
- Country of origin: European Union (Poland)

## POWER SUPPLY

AC line input

Voltage ranges: 100–240V alternating current (AC)

Frequency: 50–60Hz single phase

## 3G ANTENNA

2x Omnidirectional 3.5dBi antenna with magnetic foot

Cable length 3m

## SENDING/RECEIVING THROUGHPUT

- Incoming transmission rate: up to 60 SMS/min
- Outgoing transmission rate: up to 40 SMS/min
- API send SMS requests: 200 SMS/min (messages are queued for sending in a built-in database)

## SOFTWARE PLATFORM

- Operating system: Linux kernel 4.4
- built-in Apache2 web server
- built-in PostgreSQL database server
- built-in Postfix email server
- built-in SNMP agent
- built-in NTP-client
- built-in Failover (HA-cluster) service
- watchdog mechanism for 3G modems
- failover mechanism for built-in 3G modems
- modern responsive web interface

## 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.*



## REGULATORY STATEMENTS

---

### FCC compliance statement

---

This device complies with part 15 of the FCC rules. Operation is subject to the following two conditions:

- (1) This device may not cause harmful interference, and
- (2) this device must accept any interference received, including interference that may cause undesired operation.

**Note:**

This equipment has been tested and found to comply with the limits for a Class A device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a business/commercial non-residential environment. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment to an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

**Important:**

This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the manufacturer's instruction manual, may cause harmful interference with radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case you will be required to correct the interference at your own expense. The FCC regulations provide that changes or modifications not expressly approved by SMSEagle™ could void your authority to operate this equipment. This product has demonstrated EMC compliance under conditions that included the use of compliant peripheral devices (antennas) and shielded cables between system components. It is important that you use compliant peripheral devices and shielded cables between system components to reduce the possibility of causing interference to radios, televisions, and other electronic devices.

### Canadian regulatory statement

---

This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions:

- (1) this device may not cause interference, and
- (2) this device must accept any interference, including interference that may cause undesired operation of the device.

This Class A digital apparatus meets the requirements of the Canadian Interference-Causing Equipment Regulations.

*CAN ICES-3 (A)/NMB-3(A)*

## Disposal and recycling information

---

Your SMSEagle device contains lithium-ion battery for RTC backup. Dispose of the device and/or battery in accordance with local environmental laws and guidelines.

*European Union—Disposal Information*



The symbol above means that according to local laws and regulations your product and/or its battery shall be disposed of separately from household waste. When this product reaches its end of life, take it to a collection point designated by local authorities. The separate collection and recycling of your product and/or its battery at the time of disposal will help conserve natural resources and ensure that it is recycled in a manner that protects human health and the environment.



Ul. Piątkowska 163, 60-650  
Poznań, Poland | Europe

T +48 61 6713 413

E [hello@smseagle.eu](mailto:hello@smseagle.eu)

[www.smseagle.eu](http://www.smseagle.eu)