SDK API Manual

Version 1.3

NetModule AG, Switzerland

May 11, 2018

Contents

1 Introduction **3**

2 API functions **4**

2.1 Serial . . . 4
2.1.1 nb_serial_getattr . . . 4
2.1.2 nb_serial_setattr . . . 4
2.1.3 nb_serial_write . . . 4
2.1.4 nb_serial_read . . . 5
2.2 Media . . . 5
2.2.1 nb_media_mount . . . 5
2.2.2 nb_media_umount . . . 5
2.2.3 nb_media_getmount . . . 5
2.3 Modbus . . . 6
2.3.1 nb_modbus_register . . . 6
2.3.2 nb_modbus_unregister . . . 6
2.3.3 nb_modbus_set_slave . . . 6
2.3.4 nb_modbus_flush . . . 6
2.3.5 nb_modbus_last_error . . . 7
2.3.6 nb_modbus_set_debug . . . 7
2.3.7 nb_modbus_send_raw . . . 7
2.3.8 nb_modbus_reply_raw_response . . . 7
2.3.9 nb_modbus_extract_payload . . . 8
2.3.10 nb_modbus_read_bits . . . 8
2.3.11 nb_modbus_read_regs . . . 8
2.3.12 nb_modbus_write_bits . . . 8
2.3.13 nb_modbus_write_input_bits . . . 9
2.3.14 nb_modbus_write_regs . . . 9
2.3.15 nb_modbus_write_input_regs . . . 9
2.3.16 nb_modbus_receive . . . 9
2.3.17 nb_modbus_reply . . . 10
2.4 SMS . . . 11
2.4.1 nb_sms_send . . . 11
2.4.2 nb_sms_sendmsg . . . 11

2.4.3 nb_sms_list . . . 11
2.4.4 nb_sms_retrieve . . . 12
2.4.5 nb_sms_header . . . 12
2.4.6 nb_sms_body . . . 12
2.4.7 nb_sms_delete . . . 12
2.5 E-Mail . . . 12
2.5.1 nb_email_send . . . 12
2.5.2 nb_mail_list . . . 13
2.5.3 nb_mail_delete . . . 13
2.5.4 nb_mail_fetch . . . 13
2.5.5 nb_mail_send . . . 14
2.6 Digital I/O . . . 14
2.6.1 nb_dio_get . . . 14
2.6.2 nb_dio_set . . . 14
2.6.3 nb_dio_count . . . 15
2.6.4 nb_dio_summary . . . 15
2.7 Configuration . . . 15
2.7.1 nb_config_get . . . 15
2.7.2 nb_config_set . . . 15
2.7.3 nb_config_summary . . . 16
2.8 Status Information . . . 16
2.8.1 nb_status . . . 16
2.8.2 nb_status_summary . . . 17
2.9 Network Scanning . . . 18
2.9.1 nb_scan_networks . . . 18
2.10 File Transfers . . . 18
2.10.1 nb_transfer_get . . . 18
2.10.2 nb_transfer_put . . . 18
2.10.3 nb_transfer_post . . . 19
2.10.4 nb_transfer_list . . . 19
2.10.5 nb_transfer_delete . . . 20
2.11 LED . . . 20
2.11.1 nb_led_acquire . . . 20
2.11.2 nb_led_release . . . 20
2.11.3 nb_led_set . . . 21
2.12 Config/Software Update . . . 21
2.12.1 nb_update_status . . . 21
2.12.2 nb_update_config . . . 22
2.12.3 nb_update_software . . . 22
2.12.4 nb_update_sshkeys . . . 22
2.13 Web Pages . . . 22
2.13.1 nb_page_register . . . 23

2.13.2 nb_page_unregister . . . 23
2.13.3 nb_page_request . . . 23
2.13.4 nb_page_respond . . . 23
2.13.5 nb_page_finish . . . 24
2.14 Voice . . . 24
2.14.1 nb_voice_event . . . 25
2.14.2 nb_voice_endpoint_list . . . 26
2.14.3 nb_voice_endpoint_get . . . 26
2.14.4 nb_voice_call_list . . . 26
2.14.5 nb_voice_call_get . . . 26
2.14.6 nb_voice_call_dial . . . 27
2.14.7 nb_voice_call_accept . . . 27
2.14.8 nb_voice_call_route . . . 27
2.14.9 nb_voice_call_hangup . . . 27
2.14.10 nb_voice_call_volume . . . 28
2.15 SNMP . . . 28
2.15.1 nb_snmp_register . . . 28
2.15.2 nb_snmp_link . . . 28
2.15.3 nb_snmp_update . . . 29
2.15.4 nb_snmp_listen . . . 29
2.15.5 nb_snmp_unlink . . . 29
2.15.6 nb_snmp_host . . . 29
2.15.7 nb_snmp_get . . . 30
2.15.8 nb_snmp_set . . . 30
2.15.9 nb_snmp_traphost . . . 30
2.15.10 nb_snmp_trap . . . 31
2.16 CAN . . . 31
2.16.1 nb_can_setattr . . . 31
2.16.2 nb_can_open . . . 31
2.16.3 nb_can_close . . . 32
2.16.4 nb_can_setfilter . . . 32
2.16.5 nb_can_sendonly . . . 32
2.16.6 nb_can_recvmsg . . . 33
2.16.7 nb_can_sendmsg . . . 33
2.17 Network . . . 33
2.17.1 nb_gethostbyname . . . 33
2.17.2 nb_ifc_address . . . 34
2.17.3 nb_ping . . . 34
2.17.4 nb_arp_ping . . . 34
2.17.5 nb_arp_gratuitous . . . 34
2.17.6 nb_etherwake . . . 35
2.18 Other . . . 35

2.18.1 nb_syslog 35
2.18.2 nb_event_get 35
2.18.3 nb_event_msg 35
2.18.4 nb_reboot 36
2.18.5 nb_restart 36
2.18.6 nb_reset_factory 36
2.18.7 nb_reset_statistics 36

1 Introduction

This manual describes the SDK API functions which provide a range of general-purpose extensions for the Arena scripting language.
The current version ships with the following features:

```
- Send and retrieve SMS
- Send E-mail
- Read or write from or to a serial device
- Control digital input/output ports
- Run TCP/UDP servers
- Run IP/TCP/UDP clients
- Access files of mounted media (e.g. an USB stick)
- Retrieve status information from the system
- Get or set configuration parameters
- Write to syslog
- Transfer files over HTTP/FTP
- Perform config/software updates
- Control LEDs
- Get system events, restart services or reboot system
- Scan for networks in range
- Create your own web pages
- Voice control functions
- SNMP functions
- CAN socket functions
- Various network-related functions
- Other system-related functions
```

2 API functions

2.1 Serial

2.1.1 nb_serial_getattr

```
struct nb_serial_getattr (string dev)
```

The nb_serial_getattr function retrieves the current attributes of a serial device.

dev	serial device (e.g. serial0 for first device)

Returns a struct containing values for baudrate, databit, stopbit, parity, flowctl or void on error.

2.1.2 nb_serial_setattr

```
int nb_serial_setattr (string dev, int b, int d, int s, int p, int f)
```

The nb_serial_setattr function can be used to set the attributes of a serial device.

dev	serial device (e.g. serial0 for first device)
b	baudrate (e.g. 9600, 19200, 38400, 57600, 115200)
d	number of data bits (5, 6, 7, 8)
s	number of stop bits (1, 2)
p	parity (0=no parity, 1=odd parity, 2=even parity)
f	flow control (0=none, 1=xon/xoff, 2=hardware)

Returns -1 on error, otherwise zero.

2.1.3 nb_serial_write

```
int nb_serial_write (string dev, string msg)
```

The nb_serial_write function can be used for writing a message directly to a serial device.

dev	serial device (e.g. serial0 for first device)
msg	message to be written

Returns number of bytes write or -1 on error.

2.1.4 nb_serial_read

```
string nb_serial_read (string dev)
```

The nb_serial_read function can be used to read a message from a serial device.

dev	serial device (e.g. serial0 for first device)

Returns the string received from the serial port or an empty string in case of errors.

2.2 Media

2.2.1 nb_media_mount

```
int nb_media_mount (string dev)
```

The nb_media_mount function mounts the specified media device.

dev	device name (eg. usb0)

Returns 0 on success and -1 on error. The media will be mounted to /mnt/media/usb0 for instance. You may use any IO functions afterwards to operate on it.

2.2.2 nb_media_umount

```
int nb_media_umount (string dev)
```

The nb_media_umount function umounts the specified media device.

dev	device name (e.g usb0)

Returns -1 on error.

2.2.3 nb_media_getmount

```
string nb_media_getmount (void)
```

The nb_media_getmount function returns a list of any currently mounted media including the corresponding mountpoint (i.e. in the form "<media> on <path>"). If nothing is mounted (or in case of an error) an empty string will be returned.

2.3 Modbus

2.3.1 nb_modbus_register

```
int nb_modbus_register (int fd, int type)
```

This function will register a file descriptor (as returned by open or accept) to the modbus subsystem.

fd	file descriptor
type	can be either MODBUS_TYPE_TCP or MODBUS_TYPE_RTU

On success, the function returns 0. Otherwise -1.

2.3.2 nb_modbus_unregister

```
int nb_modbus_unregister (int fd)
```

This function unregisters a previously registered file descriptor.

fd	file descriptor

On success, the function returns 0. Otherwise -1.

2.3.3 nb_modbus_set_slave

```
int nb_modbus_set_slave (int fd, int slave)
```

The nb_modbus_set_slave function applies the local slave identifier number which is required when communicating in RTU mode.

fd	file descriptor
slave	slave identifier

The function will return zero if successful. Otherwise it returns -1, the error can be figured out using nb_modbus_last_error.

2.3.4 nb_modbus_flush

```
int nb_modbus_flush (int fd)
```

The nb_modbus_flush function will discard any data received without reading from the file descriptor.

fd	file descriptor

The function will return zero or the number of flushed bytes in case of success. Otherwise it returns -1, the error can be figured out using nb_modbus_last_error.

2.3.5 nb_modbus_last_error

```
string nb_modbus_last_error (void)
```

The nb_modbus_last_error function show the last occurred error.

2.3.6 nb_modbus_set_debug

```
int nb_modbus_set_debug (int fd, bool flag)
```

The nb_modbus_set_debug function enables or disables the debug mode.

fd	file descriptor
flag	true for enabled or false for disabled

The function will return zero on success, otherwise -1.

2.3.7 nb_modbus_send_raw

```
array nb_modbus_send_raw (int fd, array request)
```

The nb_modbus_send_raw function sends the request to the associated descriptor and receives the confirmation.

fd	file descriptor
request	modbus raw request

The functions returns the modbus confirmation if successful. Otherwise it will return void.

2.3.8 nb_modbus_reply_raw_response

```
int nb_modbus_reply_raw_response (int fd, array response)
```

The nb_modbus_reply_raw_response function will reply to a modbus request.

fd	file descriptor
response	the raw modbus response

The nb_modbus_replay_raw_response function will return the number of bytes sent on succcess. Otherwise it will return -1.

2.3.9 nb_modbus_extract_payload

```
array nb_modbus_extract_payload (int fd, array request)
```

The nb_modbus_extract_payload function extracts the payload from a given request.

fd	file descriptor
request	modbus request

It returns the extracted payload from the request if successful, otherwise void.

2.3.10 nb_modbus_read_bits

```
array nb_modbus_read_bits (int fd, int addr, int len)
```

The nb_modbus_read_bits function reads the status of the bits from the remote device.

fd	file descriptor
addr	address of bits to read
len	length of data to read

The function returns the number of read status bits/registers if successful, otherwise it returns -1.

2.3.11 nb_modbus_read_regs

```
array nb_modbus_read_regs (int fd, int addr, int len)
```

The nb_modbus_read_regs function reads the status of the registers from the remote device.

fd	file descriptor
addr	address of registers to read
len	length of data to read

The function returns the number of read status registers if successful, otherwise -1.

2.3.12 nb_modbus_write_bits

```
int nb_modbus_write_bits (int fd, int addr, int length, array data)
```

The nb_modbus_write_bits function writes the status of bits to the remote device.

fd	file descriptor
addr	address of bits to write
length	length of array
data	array to write

The function returns the number of bits written if successful, otherwise -1.

2.3.13 nb_modbus_write_input_bits

```
int nb_modbus_write_input_bits (int fd, int addr, int length, array data)
```

The nb_modbus_write_bits function writes the status of input bits to the remote device.

fd	file descriptor
addr	address of input bits to write
length	length of array
data	array to write

The function shall return the number of written bits if successful, otherwise -1.

2.3.14 nb_modbus_write_regs

```
int nb_modbus_write_regs (int fd, int addr, int length, array data)
```

The nb_modbus_write_regs function writes the status of the registers to the remote device.

fd	file descriptor
addr	address of registers to write
length	length of array
data	array to write

The function returns the number of written bits if successful, otherwise -1.

2.3.15 nb_modbus_write_input_regs

```
int nb_modbus_write_input_regs(int fd, int addr, int length, array data)
```

The nb_modbus_write_input_regs function writes the status of the input registers to the remote device.

fd	file descriptor
addr	address of input registers to write
length	length of array
data	array to write

The function returns the number of written bits if successful, otherwise -1.

2.3.16 nb_modbus_receive

```
int nb_modbus_receive (int fd)
```

The nb_modbus_receive function will receive an indication request from the specified descriptor. This function is used by a modbus slave/server to receive and analyze indication requests sent by the masters/clients.

fd	file descriptor

The function returns the received indication request.

2.3.17 nb_modbus_reply

```
int modbus_reply (int fd, array req, struct resp)
```

The nb_modbus_reply function sends a response for a received request (as returned by nb_modbus_receive) to the specified descriptor.

fd	file descriptor
req	request
resp	response struct made up as follows:

```
resp = mkstruct(
   "bits", mkarray
    (
        0, 0, 0, 1, 1, 1, 1,
        0, 0, 0, 1, 1, 1, 1
    ),
    "ibits", mkarray
    (
       1, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1
    ),
    "regs", mkarray
    (
        0x0000  0x0001, 0x0002, 0x0003, 0x0004, 0x0005, 0x0006, 0x0007,
        0x0008, 0x0009, 0x000A, 0x000B, 0x000C, 0x000D, 0x000E, 0x000F
    ),
    "iregs", mkarray
    (
        0xFF00, 0xFF01, 0xFF02, 0xFF03, 0xFF04, 0xFF05, 0xFF06, 0xFF07,
        0xFF08, 0xFF09, 0xFF0A, 0xFF0B, 0xFF0C, 0xFF0D, 0xFF0E, 0xFF0F
    )
);
```

```
Representation:
```

```
"bits"  => Discrete Output Coils
"ibits" => Discrete Input Contacts
"regs"  => Analog Output Holding Registers
"iregs" => Analog Input Registers
```

2.4 SMS

Please note that the SMS daemon must be properly configured prior to using the functions below.

2.4.1 nb_sms_send

```
string nb_sms_send (string number, string msg)
```

The nb_sms_send function can be used to send an SMS to the specified number.

number	recipient's phone number (international format)
msg	the message to be sent

Returns the resulting message identifier on success or an empty string on error.

2.4.2 nb_sms_sendmsg

string nb_sms_sendmsg (struct msg)
The nb_sms_send function can be used to send an SMS with parameters specfied in the struct msg which includes the following fields:

number	recipient's phone number (international format)
report	request delivery report (if set to yes)
gateway	the SMS gateway used for sending the message

modem the modem over which the message shall be sent

msg	the message to be sent

Returns the resulting message identifier on success or an empty string on error.

2.4.3 nb_sms_list

```
array nb_sms_list (void)
```

The nb_sms_list function can be used to retrieve the list of messages in the inbox. Returns an array of message identifiers.

2.4.4 nb_sms_retrieve

```
string nb_sms_retrieve (string id)
```

The nb_sms_retrieve function returns the message text of the specified message identifier.

id	the message identifier

2.4.5 nb_sms_header

```
string nb_sms_header (string id, string tag)
```

The nb_sms_header function returns the headers of a given message identifier.

id	the message identifier
tag	a specific header tag (such as "From")

Returns the value of the specified header tag or all headers (if tag omitted) or an empty string on error.

2.4.6 nb_sms_body

```
string nb_sms_body (string id)
```

The nb_sms_body function returns the body of a given message identifier.

id	the message identifier

Returns the message's body text or an empty string on error.

2.4.7 nb_sms_delete

```
int nb_sms_delete (string id)
```

The nb_sms_delete function can be used to delete a message from the inbox.

id	the message identifier

Returns zero on success or -1 on error.

2.5 E-Mail

2.5.1 nb_email_send

```
int nb_email_send (string rcpt, string subj, string msg)
```

The nb_email_send function can be used to send an E-Mail to a particular address.

rcpt	recipient's email address (e.g. abc@abc.com)
subj	email subject
msg	email content

Returns zero on success or any error code. Please note that the E-Mail client must be properly configured prior to using this function.

2.5.2 nb_mail_list

```
int nb_mail_list (string usr, string pwd, string url)
```

The nb_mail_list function can be used to get the number of existing mails at a remote IMAP/POP3 server.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	IMAP/POP3 server URL (e.g. imap://mail.example.com)

Returns number of available mails or -1 on error. Please note that IMAP functions are limited to the INBOX folder.

2.5.3 nb_mail_delete

```
int nb_mail_delete (string usr, string pwd, string url, int index)
```

The nb_mail_delete function can be used to delete an E-Mail from a remote IMAP/POP3 server.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	IMAP/POP3 server URL (e.g. imap://mail.example.com)
index	the mail index to be deleted

Returns 0 on success.

2.5.4 nb_mail_fetch

```
int nb_mail_fetch (string usr, string pwd, string url, int index)
```

The nb_mail_fetch function can be used to fetch an E-Mail from a remote IMAP/POP3 server.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	IMAP/POP3 server URL (e.g. imap://mail.example.com)
index	the mail index to be fetched

Returns void on error, otherwise a struct with the following fields:

from	sender's address
to	recipient's address
subject	subject of the mail
date	when the mail has been sent
body	content of the mail

2.5.5 nb_mail_send

```
int nb_mail_send (string usr, string pwd, string url, struct mail)
```

The nb_mail_send function can be used to send an E-Mail via a remote SMTP server.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	SMTP server URL (e.g. smtp://mail.example.com)
mail	a struct containing the fields from, to, subject and body

Return -1 on error, otherwise zero.

2.6 Digital I/O

2.6.1 nb_dio_get

```
int nb_dio_get (string port)
```

The nb_dio_get function retrieves the status of a digital I/O port.

port	DIO port to be queried (in1, in2, out1, out2)

Returns the DIO status (0 = off, 1 = on) or -1 on error.

2.6.2 nb_dio_set

```
int nb_dio_set (string port, int state)
```

The nb_dio_set function can be used to turn on/off the status of a digital output port.

port	digital output port to be configured (out1, out2)
state	new output status (0 = off, 1 = on)

Returns -1 on error.

2.6.3 nb_dio_count

```
int nb_dio_count (string port)
```

The nb_dio_count function can be used to get the number of toggles of the specified input port.

port	digital input port (in1, in2)

Returns the number of toggles since the last measurement.

2.6.4 nb_dio_summary

```
string nb_dio_summary (void)
```

The nb_dio_summary function retrieves the status of all digital I/O ports.
Returns a string holding the status of all ports or an empty string on error.

2.7 Configuration

2.7.1 nb_config_get

```
string nb_config_get (string key)
```

The nb_config_get function returns the currently configured value of a particular config parameter.

key	config key (e.g. "config.info")

Returns the config value or an empty string on error.

2.7.2 nb_config_set

```
int nb_config_set (string config)
```

The nb_config_set function can be used to set system configuration parameters.

config	config to be set in the form key=value (e.g. sdk.status=0)

Returns -1 on error. The config values will be immediately applied to the system.

2.7.3 nb_config_summary

```
string nb_config_summary (void)
```

The nb_config_summary function returns the current system configuration which corresponds to the delta of the factory configuration and the currently active configuration.

2.8 Status Information

2.8.1 nb_status

```
struct nb_status (string section)
```

The nb_status function will return various status values (as available through cli).

section	the status section which shall be queried

The following sections can be specified:

info	System and config information
config	Current configuration
system	System information
configuration	Configuration information
license	License information
wwan	WWAN module status
wlan	WLAN module status
gnss	GNSS (GPS) module status
eth	Ethernet interface status
lan	LAN interface status
wan	WAN interface status
openvpn	OpenVPN connection status
ipsec	IPsec connection status
pptp	PPTP connection status
gre	GRE connection status
dialin	Dial-In connection status
mobileip	MobileIP status
dio	Digital IO status
audio	Audio module status
can	CAN module status
uart	UART module status
ibis	IBIS module status
redundancy	Redundancy status
sms	SMS status
firewall	Firewall status
qos	QoS status
neigh	Neighborhood status
location	Current Location

Returns a struct holding the relevant status values (see 'status.are' example).

2.8.2 nb_status_summary

```
string nb_status_summary (void)
```

The nb_status_summary function will return a short summary about the current system status or an empty string on error.

2.9 Network Scanning

2.9.1 nb_scan_networks

```
struct nb_scan_networks (string ifc)
```

The nb_scan_networks function can be used to scan for available networks.

ifc	the interface to scan (e.g. WLAN1 or Mobile1)

Returns a struct holding the relevant networks (see examples).

Please note that scanning a mobile interface will tear down any running WWAN connections. Same applies to WLAN interfaces operating in access-point mode. Therefore the scan interval is limited to 30 seconds.

2.10 File Transfers

The file transfer functions can be used to transfer files from or to a remote server denoted by an FTP or HTTP/HTTPS URL. Please note that all functions operate on files in the SDK sandbox (which is /mnt/sdk on the host system).

2.10.1 nb_transfer_get

```
int nb_transfer_get (string usr, string pwd, string url, string path)
```

The nb_transfer_get function can be used to get a file from a remote server. If both, username and password are specified, the function will perform authentication based on the relevant methods of HTTP or FTP.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	the URL where to get the file from
path	the local path where the file should be stored

Returns -1 on error.

2.10.2 nb_transfer_put

```
int nb_transfer_put (string usr, string pwd, string url, string path)
```

The nb_transfer_put function can be used to transfer a file to a remote server. The usr/pwd arguments can be applied in order to perform authentication.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	the URL where to put the file to
path	the path to the local file which sould be sent

Returns -1 on error.

2.10.3 nb_transfer_post

```
int nb_transfer_post (string usr, string pwd, string url, string path,
    string pp)
```

The nb_transfer_post function can be used to transfer a file to a remote HTTP server. By using the POST method, additional parameters may be passed. The usr/pwd arguments can be applied in order to perform authentication.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	the URL where to put the file to
path	the path to the local file which sould be sent
pp	additional POST parameters

Returns -1 on error.

POST parameters have to be provided as follows:

```
<key1>=<val1>&<key2>=<val2>&<keyN>=<valN>
```

2.10.4 nb_transfer_list

```
array nb_transfer_list (string usr, string pwd, string url)
```

The nb_transfer_list function can be used to retrieve the list of files from a remote server. The usr/pwd arguments can be applied in order to perform authentication.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	the URL specifying the directory to be listed

Returns an array of struct describing the directory files. They are made up of:

name	name of the file
size	file size in bytes
mode	file mode and permission (see chmod)
user	owner username
group	owner groupname
time	modification time
tm	time struct of modification time

2.10.5 nb_transfer_delete

```
int nb_transfer_delete (string usr, string pwd, string url)
```

The nb_transfer_delete function can be used to delete a file from a remote FTP server. The usr/pwd arguments can be applied in order to perform authentication.

usr	the username used for authentication (can be empty)
pwd	the password used for authentication (can be empty)
url	the URL specifying the path of the file to be deleted

Returns -1 on error.

2.11 LED

2.11.1 nb_led_acquire

```
int nb_led_acquire (int led)
```

The nb_led_acquire function will acquire the specified LED for a particular script. Any associated system indication on that LED will be stopped until the LED is released again.

led	the LED number to be acquired (starting from left/top) or LED_ALL for all LEDs

Returns -1 on error, otherwise zero. Please note that the status LED cannot be acquired.

2.11.2 nb_led_release

```
int nb_led_release (int led)
```

The nb_led_release function will release an acquired LED again.

led	the LED number to be released or LED_ALL for all LEDs

Returns -1 on error, otherwise zero.

2.11.3 nb_led_set

```
int nb_led_set (int led, int mode)
```

The nb_led_set function will set the specified LED to a specific mode.

led	the LED number to be released or LED_ALL for all LEDs
mode	the LED mode to be applied

Returns -1 on error, otherwise zero.

LED modes can be specified by OR'ing the following colors and types:

```
LED_OFF                 turn off LED
LED_COLOR_GREEN         color is green
LED_COLOR_RED           color is red
LED_COLOR_YELLOW        color is yellow
LED_SOLID               type is solid
LED_BLINK_FAST          type is fast blinking
LED_BLINK_SLOW          type is slow blinking
```

2.12 Config/Software Update

The following functions can be used to trigger a configuration or software update of the system. The URL can be specified as follows:

```
http://<server>/<path>      (retrieve file via HTTP)
https://<server>/<path>     (retrieve file via HTTPS)
ftp://<server>/<path>       (retrieve file via FTP)
tftp://<server>/<path>      (retrieve file via TFTP)
file:///<path>              (use local file)
```

Please bear in mind that calling nb_update_software will result in a system reboot. The nb_update_config call will restart the SDK which will terminate your scripts. Thus, it is recommended to exit the script after calling this function and check the result later on via nb_update_status. If a file URL is used, the path must correspond to an absolute path to the root directory. Using /tmp for update tasks is currently not possible.

2.12.1 nb_update_status

```
string nb_update_status (void)
```

The nb_update_status function returns the status of the last or currently running update operation

2.12.2 nb_update_config

```
int nb_update_config (string url)
```

The nb_update_config function will perform a configuration update from the specified URL.

url	the URL of the config file

Returns zero on success.

Please note that any running SDK script will be terminated during the update process. Thus, the script must exit after nb_update_config() has been called.

2.12.3 nb_update_software

```
int nb_update_software (string url)
```

The nb_update_software function will perform a software update from the specified URL.

url	the URL of the software image

Returns zero on success.

2.12.4 nb_update_sshkeys

```
int nb_update_sshkeys (string url)
```

The nb_update_sshkeys function will perform an update of the SSH authorized keys.

url	the URL of the keys file

Returns zero on success.

2.13 Web Pages

The following functions can be used to implement your own pages within the Web Manager. Such a page will appear under the SDK menu as soon as it has been registered.

2.13.1 nb_page_register

```
int nb_page_register (int id, string title)
int nb_page_register (int id, string title, string submenu)
```

The nb_page_register function registeres a new page with the specified identifier and title. If submenu is specified it will be hooked into the specified menu.

id	identifier
title	page title
submenu	submenu for page

Returns -1 on error, otherwise a page structure which can be used for other page functions.

2.13.2 nb_page_unregister

```
int nb_page_unregister (struct page)
```

The nb_page_unregister function can be used to unregister a page again.

page	page structure

Returns -1 on error, otherwise zero.

2.13.3 nb_page_request

```
struct nb_page_request (struct page)
```

The nb_page_request function listens for incoming requests.

page	page structure

Returns void on error, otherwise a request structure which holds possible GET and POST parameters.

2.13.4 nb_page_respond

```
int nb_page_respond (struct page, string fmt, ...)
```

The nb_page_respond function can be used to echo back a string to the request and can be called multiple times until nb_page_finish is called. It supports a format string and additional arguments that are formatted accordingly. Please refer to the printf function for more information about formatting options.

page	page structure
fmt	format string

Returns -1 on error and zero on success.

2.13.5 nb_page_finish

```
int nb_page_finish (struct page)
```

The nb_page_finish function can be used to finish a request. Any data will be passed to the client then.

page	page structure

Returns -1 on error and zero on success.

2.14 Voice

The voice control functions mentioned below can be used to control the behaviour of the voice gateway which is responsible for dispatching calls between Voice-Over-Mobile, SIP and Audio endpoints.
Calls are represented as structures which may look like:

```
struct(5): {
  .id = int: 12345
  .state = string[7]: "dialing"
  .calling = string[24]: "sip://user@192.168.1.254:5060"
  .called = string[22]: "vom://+123456789@Vom1"
};
```

The following states are possible:

```
routing          call is in routing state
dialing          call is in dialing state
alerting         call is in alerting state
active           call is active
hungup           call had hung up
```

In common, the functions can operate with either a call identifier or the call struct itself (e.g. if further parameters need to be provided).
Endpoints are represented as structures which may look like:

```
struct(3): {
  .id = int: 54321
  .desc = string[5]: "vom://Vom1"
```

```
  .state = string[4]: "busy"
  .volume = = int: 7
};
```

Endpoints can be specified by ID or a descriptor which can be made up as follows:

Sip1	First SIP subscriber
Vom1	First Voice-Over-Mobile
Aud1	First Audio device

The following URLs are valid descriptors as well:

```
54321                           endpoint ID
vom://++123                     Voice-Over-Mobile number
vom://++123@Vom1                Voice-Over-Mobile number on Vom1
sip://user@192.168.1.254:5060   SIP address
sip://user                      SIP user (must be subscribed)
aud://Aud1                      Audio device
```

The following states are possible:

```
busy            endpoint is already holding a call
available       endpoint is ready to take a call
```

2.14.1 nb_voice_event

```
struct nb_voice_event (int timeout)
```

The nb_voice_event function listens for any new voice events.

```
timeout         timeout in seconds
```

Returns void on error, otherwise a structure holding the event type and the according call:

```
struct(2): {
  .type = string[8]: "dispatched"
  .call = struct(5): {
     .id = int: 12345
     .state = string[7]: "alerting"
     .calling = string[24]: "sip://user@192.168.1.254:5060"
     .called = string[22]: "vom://+123456789@Vom1"
   }
};
```

The following event types are possible:

```
incoming        call is coming in from calling endpoint (ready to route)
outgoing        call is going out to calling endpoint (ready to route)
dialing         call is dialing the called endpoint
dispatched      call has been dispatched (alerting the called endpoint)
connected       call is connected to the called endpoint
hungup          call has hung up
```

2.14.2 nb_voice_endpoint_list

```
array nb_voice_endpoint_list (void)
```

The nb_voice_endpoint_list function lists all currently known endpoints. Returns void on error, otherwise an array holding the endpoint structures.

2.14.3 nb_voice_endpoint_get

```
struct nb_voice_endpoint_get (endpoint)
```

The nb_voice_endpoint_get function can be used to lookup or update a specific endpoint.

endpoint	endpoint struct, ID or descriptor

Returns void on error, otherwise the corresponding endpoint structure.

2.14.4 nb_voice_call_list

```
array nb_voice_call_list (void)
```

The nb_voice_call_list function lists all currently known calls. Returns void on error, otherwise an array holding the call structures.

2.14.5 nb_voice_call_get

```
struct nb_voice_call_get (call)
```

The nb_voice_call_get function can be used to lookup or update a specific call.

```
call    call struct or id
```

Returns void on error, otherwise the corresponding call structure.

2.14.6 nb_voice_call_dial

```
int nb_voice_call_dial (call)
```

The nb_voice_call_dial function can be used to dial a new call.

```
call    call struct
```

Returns -1 on error, otherwise the corresponding result.

2.14.7 nb_voice_call_accept

```
int nb_voice_call_accept (call)
```

The nb_voice_call_accept function can be used to accept calls in dispatch state.

```
call    call struct or id
```

Returns -1 on error, otherwise the result.
Remark: This function can be used to take a call for audio endpoints.

2.14.8 nb_voice_call_route

```
int nb_voice_call_route (call, endpoint)
```

The nb_voice_call_route function can be used to route incoming or outgoing calls to a dedicated endpoint.

```
call            call struct or id
endpoint        endpoint struct, ID or descriptor
```

Returns -1 on error, otherwise the result.

2.14.9 nb_voice_call_hangup

```
int nb_voice_call_hangup (call)
```

The nb_voice_call_hangup function can be used to hangup or drop a call.

```
call    call struct or id
```

Returns -1 on error, otherwise the result.

2.14.10 nb_voice_call_volume

```
int nb_voice_call_volume (endpoint, int level)
```

The nb_voice_call_volume function can be used to adjust the volume level of a call.

```
endpoint        endpoint struct, ID or descriptor
level           volume level (0 to 7)
```

Returns -1 on error, otherwise the result.

2.15 SNMP

The SNMP functions below offer facilities to

```
- expose certain OIDs to the SNMP agent
- extend the list of MIB entities
- run SET or GET commands
- issue SNMP traps
```

Only integer and octet string entities are currently supported.

2.15.1 nb_snmp_register

```
int nb_snmp_register (string name, int ext, string type, string mode)
```

The nb_snmp_register function will register a MIB entity.

```
name    name of entity
ext     the OID extension number of the entity
type    type of entity (i for integer, s for octet string)
mode    mode of entity
```

Returns -1 on error. Please note that only scalars are currently supported.

2.15.2 nb_snmp_link

```
int nb_snmp_link (void)
```

The nb_snmp_link function will link any registered MIB entities to the agent. The entities will be accessible from an SNMP client over .1.3.6.1.4.1.<vendor>.10.90 after this function has been called. The default values are 0 for integers and an empty string for octet strings.

Returns -1 on error.

2.15.3 nb_snmp_update

```
int nb_snmp_update (string name, string value)
```

The nb_snmp_update function will update the specified MIB entity to the given value.

name	name of entity
value	value to be set

Returns -1 on error.

2.15.4 nb_snmp_listen

```
int nb_snmp_listen (int timeout)
```

By using the nb_snmp_listen function it is possible to get notified as soon as an entity has been set by a client.

timeout	timeout to wait in seconds

Returns a struct containing the name and value of the set entity. Otherwise, void will be returned

2.15.5 nb_snmp_unlink

```
int nb_snmp_unlink (void)
```

The nb_snmp_unlink function disconnects any MIB entities from the agent. Returns -1 on error.

2.15.6 nb_snmp_host

```
int nb_snmp_host (string host, int port, int version, string community)
int nb_snmp_host (string host, int port, int version, string user,
    string password, string auth, string priv)
```

The nb_snmp_host function will set the SNMP host for running SET or GET requests. For an SNMPv1/v2 host the parameters are:

host	hostname or address
port	trap port
version	SNMP version (1 or 2)
community	community string

For an SNMPv3 host the parameters are:

host	hostname or address
port	port
version	SNMP version (3)
user	username
pass	password
auth	authentication protocol (MD5 or SHA)
priv	privacy protocol (DES or AES)

Returns -1 on error.

2.15.7 nb_snmp_get

```
void nb_snmp_get (string oid)
```

The nb_snmp_get function will perform a GET request for the specified OID. An SNMP host must be configured prior to using that function.

oid	the queried OID

This function returns void in case an error occured, an integer value if OID represent an integer or a string value if OID represents an octet string.

2.15.8 nb_snmp_set

```
int nb_snmp_set (string oid, string type, string value)
```

The nb_snmp_set function will perform a SET request for the specified OID. An SNMP host must be configured prior to using that function.

```
oid     the OID to be set
type    the OID type ("i" for integer or "s" for octet string)
value   the value to be set
```

Returns -1 on error.

2.15.9 nb_snmp_traphost

```
int nb_snmp_traphost (string host, int port, int version, string
    community)
int nb_snmp_traphost (string host, int port, int version, string user,
    string password, string auth, string priv)
```

The nb_snmp_traphost function will set the host for sending SNMP traps. The same parameters as for nb_snmp_host apply. Returns -1 on error.

2.15.10 nb_snmp_trap

```
string nb_snmp_trap (string oid, string type, string value)
```

The nb_send_trap function will send an SNMP trap with the specified OID to a remote traphost.

oid	SNMP object identifier of the trap
type	type of value to be sent ('e' for empty, 'i' for integer and 's' for octet string)
value	value to be sent

Please note that a traphost has to be set prior to using this function.

Returns -1 on error.

2.16 CAN

The following functions can be used to communicate with the CAN interface.

2.16.1 nb_can_setattr

```
int nb_can_setattr (string ifc, int bitrate, int listenonly, int restart)
```

The nb_can_setattr function can be used to set the attributes of a CAN interface.

ifc	name of interface (e.g. can0)
bitrate	bitrate (e.g. 500000)
listenonly	sets ctrlmode listenonly
restart	restart timeout in case of a bus-off (0 = disabled)

Returns -1 on error, otherwise zero.

2.16.2 nb_can_open

```
int nb_can_open (string ifc)
```

ifc	name of interface (e.g. can0)

The nb_can_open function enables the specified interface and returns a raw socket descriptor. Please note that the attributes (e.g. bitrate) have to be set in advance before opening any interface.

Returns -1 on error.

2.16.3 nb_can_close

```
int nb_can_open (int socket)
```

socket	socket descriptor

The nb_can_close function closes the specified socket descriptor and disables the associated interface. Returns -1 on error.

2.16.4 nb_can_setfilter

```
int nb_can_setfilter (int socket, int id, int mask)
```

The nb_can_setfilter function can be used to specify which CAN frames shall be filtered out and which shall be passed to the upper layers.

socket	socket descriptor
id	CAN filter ID
mask	CAN filter mask

Returns -1 on error.

A filter matches if received-id & mask == id & mask. The filter can also be inverted (CAN_INV_FILTER bit set in id) or it can filter for error frames (CAN_ERR_FLAG bit set in mask).

2.16.5 nb_can_sendonly

```
int nb_can_sendonly (int socket)
```

The nb_can_sendonly function can be used to disable the reception of CAN frames on the selected socket.

socket	socket descriptor

Returns -1 on error.

2.16.6 nb_can_recvmsg

```
struct nb_can_recvmsg (int socket, int timeout)
```

socket	socket descriptor
timeout	timeout to wait for a message (0 means infinite)

The nb_can_recvmsg function can be used to receive a raw message from the CAN bus. Returns void on error, otherwise it returns a msg struct containing the fields:

id	32 bit CAN ID + EFF/RTR/ERR flags
data	received data (max. 8 bytes0

The ID can be examined using the following bit operators:

```
CAN_EFF_FLAG    EFF/SFF is set in the MSB
CAN_RTR_FLAG    remote transmission request
CAN_ERR_FLAG    error frame
CAN_SFF_MASK    standard frame format (SFF)
CAN_EFF_MASK    extended frame format (EFF)
CAN_ERR_MASK    omit EFF, RTR, ERR flags
```

2.16.7 nb_can_sendmsg

```
int nb_can_sendmsg (int socket, struct msg)
```

socket	socket descriptor
msg	message struct (id + data)

The nb_can_recvmsg function can be used to send a raw message to the CAN bus. Returns -1 on error, otherwise zero.

2.17 Network

2.17.1 nb_gethostbyname

```
array nb_gethostbyname (string host)
```

The nb_gethostbyname function performs a DNS lookup for the given hostname and returns an array of resolved IP addresses.

host	the to be resolved host

Returns an empty array if host could not be resolved. Please note that a valid DNS server must be available when using this function.

2.17.2 nb_ifc_address

```
string nb_ifc_address (string interface)
```

The nb_ifc_address function can be used to obtain the first address of an interface.

interface	internal interface name (e.g. lan0)

Returns the interface address or an empty string on error.

2.17.3 nb_ping

```
int nb_ping (string host)
int nb_ping (string host, int timeout)
```

The nb_ping function will send ICMP ping packets to the specified host and returns whether the host correctly responded or not.

host	the host to ping
timeout	timeout waiting for a reply (in milliseconds)

Returns 1 in case the host is alive, 0 if down and -1 on error.

2.17.4 nb_arp_ping

```
int nb_arp_ping (string host)
```

The nb_arp_ping function will send an ARP request for the specified host and returns whether the host address has been successfully resolved.

host	the host address to ping

Returns 1 in case the specified host could be resolved, 0 if not and -1 on error.

2.17.5 nb_arp_gratuitous

```
int nb_arp_gratuitous (string ifc)
int nb_arp_gratuitous (string ifc, string host)
```

The nb_arp_gratuitous function will send an gratuitous ARP advert for the address of the specified interface (or the host address provided). This can be used to update the ARP tables of your neighbors.

ifc	the interface on which the packet should be sent
host	the host address to advert

Returns 1 in case the packet has been sent or -1 on error.

2.17.6 nb_etherwake

```
int nb_etherwake (string hwaddr, string ifc)
```

The nb_etherwake function will send a WakeOnLan magic packet to wake up sleeping hosts.

hwaddr	the Ethernet MAC address of the host
ifc	the interface on which the packet is sent

Returns 0 in case the packet has been successfully sent or -1 on error.

2.18 Other

2.18.1 nb_syslog

```
int nb_syslog (string fmt, ...)
```

The nb_syslog function creates a message in the system log. Please refer to sprintf for more information about the format string and additional arguments.

msg	message to be written to syslog

Returns -1 on error.

2.18.2 nb_event_get

```
string nb_event_get (int timeout)
```

The nb_event_get function will poll for system events.

timeout	max. number of seconds to wait for an event

Returns the received event as string or an empty string in case the specified timeout has been reached.

2.18.3 nb_event_msg

```
struct nb_event_msg (int timeout)
```

The nb_event_msg function will poll for system events.

timeout	max. number of seconds to wait for an event

Returns void in case case the specified timeout has been reached or a struct with the event string and optional parameters:

```
struct(2): {
  .event = string: "call-incoming"
  .param = string[10]: "+123456789"
};
```

2.18.4 nb_reboot

```
void nb_reboot (int delay)
```

The nb_reboot function will trigger a system reboot.

delay	the delay in seconds

2.18.5 nb_restart

```
int nb_restart (string service)
```

The nb_restart function will restart the specified service.

service	the service to be restarted

Returns -1 on error, otherwise zero.

2.18.6 nb_reset_factory

```
int nb_reset_factory ()
```

The nb_reset_factory function will reset the box to factory defaults. Returns -1 on error, otherwise zero. Please note that the system will reboot after this function has been called.

2.18.7 nb_reset_statistics

```
int nb_reset_statistics (string wanlink)
```

The nb_reset_statistics function will reset all statistics (e.g. link data counters).

wanlink	the WAN link to reset (e.g. wanlink0)

All interfaces will be reset if an empty wanlink (or "all" keyword) is used. Returns -1 on error, otherwise zero.