



# SDK API Manual

Version 2.5.10

Netmodule AG, Switzerland

November 23, 2022





# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>API functions</b>	<b>8</b>
2.1	Serial . . . . .	8
2.1.1	nb_serial_getattr . . . . .	8
2.1.2	nb_serial_setattr . . . . .	8
2.1.3	nb_serial_write . . . . .	8
2.1.4	nb_serial_read . . . . .	9
2.2	Media . . . . .	9
2.2.1	nb_media_mount . . . . .	9
2.2.2	nb_media_umount . . . . .	9
2.2.3	nb_media_getmount . . . . .	9
2.3	Modbus . . . . .	10
2.3.1	nb_modbus_register . . . . .	10
2.3.2	nb_modbus_unregister . . . . .	10
2.3.3	nb_modbus_set_slave . . . . .	10
2.3.4	nb_modbus_flush . . . . .	11
2.3.5	nb_modbus_last_error . . . . .	11
2.3.6	nb_modbus_set_debug . . . . .	11
2.3.7	nb_modbus_send_raw . . . . .	11
2.3.8	nb_modbus_reply_raw_response . . . . .	11
2.3.9	nb_modbus_extract_payload . . . . .	12
2.3.10	nb_modbus_read_bits . . . . .	12
2.3.11	nb_modbus_read_input_bits . . . . .	12
2.3.12	nb_modbus_read_regs . . . . .	12
2.3.13	nb_modbus_read_input_regs . . . . .	13
2.3.14	nb_modbus_write_bits . . . . .	13
2.3.15	nb_modbus_write_input_bits . . . . .	13
2.3.16	nb_modbus_write_regs . . . . .	14
2.3.17	nb_modbus_write_input_regs . . . . .	14
2.3.18	nb_modbus_receive . . . . .	14
2.3.19	nb_modbus_reply . . . . .	14
2.4	SMS . . . . .	16

2.4.1	nb_sms_send	16
2.4.2	nb_sms_sendmsg	16
2.4.3	nb_sms_list	16
2.4.4	nb_sms_retrieve	16
2.4.5	nb_sms_header	17
2.4.6	nb_sms_body	17
2.4.7	nb_sms_delete	17
2.5	E-Mail	17
2.5.1	nb_email_send	17
2.5.2	nb_mail_list	18
2.5.3	nb_mail_delete	18
2.5.4	nb_mail_fetch	19
2.5.5	nb_mail_send	19
2.6	Digital I/O	19
2.6.1	nb_dio_get	19
2.6.2	nb_dio_set	20
2.6.3	nb_dio_count	20
2.6.4	nb_dio_summary	20
2.7	Configuration	20
2.7.1	nb_config_get	20
2.7.2	nb_config_set	21
2.7.3	nb_config_done	21
2.7.4	nb_config_summary	21
2.8	Status Information	21
2.8.1	nb_status	21
2.8.2	nb_status_summary	22
2.9	Network Scanning	23
2.9.1	nb_scan_networks	23
2.10	USSD Queries	23
2.10.1	nb_ussd_query	23
2.11	File Transfers	23
2.11.1	nb_transfer_get	24
2.11.2	nb_transfer_put	24
2.11.3	nb_transfer_post	25
2.11.4	nb_transfer_list	26
2.11.5	nb_transfer_delete	26
2.12	LED	27
2.12.1	nb_led_acquire	27
2.12.2	nb_led_release	27
2.12.3	nb_led_set	27
2.13	Config/Software Update	28
2.13.1	nb_update_status	28

2.13.2	nb_update_config . . . . .	29
2.13.3	nb_update_software . . . . .	29
2.13.4	nb_update_sshkeys . . . . .	29
2.14	Web Pages . . . . .	30
2.14.1	nb_page_register . . . . .	30
2.14.2	nb_userpage_register . . . . .	30
2.14.3	nb_page_unregister . . . . .	30
2.14.4	nb_page_request . . . . .	30
2.14.5	nb_page_respond . . . . .	31
2.14.6	nb_page_finish . . . . .	31
2.15	Voice . . . . .	31
2.15.1	nb_voice_event . . . . .	32
2.15.2	nb_voice_endpoint_list . . . . .	33
2.15.3	nb_voice_endpoint_get . . . . .	33
2.15.4	nb_voice_call_list . . . . .	34
2.15.5	nb_voice_call_get . . . . .	34
2.15.6	nb_voice_call_dial . . . . .	34
2.15.7	nb_voice_call_accept . . . . .	34
2.15.8	nb_voice_call_route . . . . .	35
2.15.9	nb_voice_call_hangup . . . . .	35
2.15.10	nb_voice_call_volume . . . . .	35
2.16	SNMP . . . . .	35
2.16.1	nb_snmp_register . . . . .	36
2.16.2	nb_snmp_link . . . . .	36
2.16.3	nb_snmp_update . . . . .	36
2.16.4	nb_snmp_listen . . . . .	37
2.16.5	nb_snmp_unlink . . . . .	37
2.16.6	nb_snmp_host . . . . .	37
2.16.7	nb_snmp_get . . . . .	38
2.16.8	nb_snmp_set . . . . .	38
2.16.9	nb_snmp_traphost . . . . .	38
2.16.10	nb_snmp_trap . . . . .	39
2.16.11	nb_snmp_inform . . . . .	39
2.17	MQTT . . . . .	39
2.17.1	nb_mqttlib_new . . . . .	39
2.17.2	nb_mqttlib_free . . . . .	40
2.17.3	nb_mqttlib_set_protocol_version . . . . .	40
2.17.4	nb_mqttlib_set_tls . . . . .	40
2.17.5	nb_mqttlib_set_tls_insecure . . . . .	41
2.17.6	nb_mqttlib_set_user_pw . . . . .	41
2.17.7	nb_mqttlib_get_callback_connect . . . . .	42
2.17.8	nb_mqttlib_get_callback_message . . . . .	42

2.17.9	nb_mqttlib_get_callback_subscribe . . . . .	43
2.17.10	nb_mqttlib_get_callback_unsubscribe . . . . .	43
2.17.11	nb_mqttlib_connect . . . . .	44
2.17.12	nb_mqttlib_disconnect . . . . .	44
2.17.13	nb_mqttlib_publish . . . . .	45
2.17.14	nb_mqttlib_set_will . . . . .	46
2.17.15	nb_mqttlib_clear_will . . . . .	46
2.17.16	nb_mqtt_publish . . . . .	46
2.17.17	nb_mqttlib_subscribe . . . . .	47
2.17.18	nb_mqttlib_unsubscribe . . . . .	49
2.18	CAN . . . . .	49
2.18.1	nb_can_setattr . . . . .	49
2.18.2	nb_can_open . . . . .	50
2.18.3	nb_can_close . . . . .	50
2.18.4	nb_can_setfilter . . . . .	50
2.18.5	nb_can_sendonly . . . . .	51
2.18.6	nb_can_recvmsg . . . . .	51
2.18.7	nb_can_sendmsg . . . . .	51
2.19	Bluetooth . . . . .	52
2.19.1	nb_bt_dev_list . . . . .	52
2.19.2	nb_bt_dev_info . . . . .	52
2.19.3	nb_bt_dev_del . . . . .	53
2.19.4	nb_bt_ada_property_getall . . . . .	54
2.19.5	nb_bt_ada_property_set . . . . .	54
2.19.6	nb_bt_adv_property_getall . . . . .	55
2.19.7	nb_bt_adv_property_set . . . . .	55
2.19.8	nb_bt_mode . . . . .	56
2.20	Network . . . . .	57
2.20.1	nb_gethostbyname . . . . .	57
2.20.2	nb_ifc_address . . . . .	57
2.20.3	nb_ping . . . . .	57
2.20.4	nb_arp_ping . . . . .	57
2.20.5	nb_arp_gratuitous . . . . .	58
2.20.6	nb_etherwake . . . . .	58
2.21	OPC-UA . . . . .	58
2.21.1	nb_opcua_connect . . . . .	59
2.21.2	nb_opcua_browse . . . . .	59
2.21.3	nb_opcua_search . . . . .	59
2.21.4	nb_opcua_read . . . . .	59
2.21.5	nb_opcua_write . . . . .	60
2.21.6	nb_opcua_disconnect . . . . .	60
2.22	Certificates . . . . .	60

2.22.1	nb_cert_install . . . . .	60
2.22.2	nb_cert_create . . . . .	61
2.22.3	nb_cert_enroll . . . . .	61
2.22.4	nb_cert_erase . . . . .	61
2.22.5	nb_cert_read . . . . .	62
2.23	Other . . . . .	62
2.23.1	nb_syslog . . . . .	62
2.23.2	nb_syslog_p . . . . .	62
2.23.3	nb_event_get . . . . .	63
2.23.4	nb_event_msg . . . . .	63
2.23.5	nb_reboot . . . . .	63
2.23.6	nb_restart . . . . .	64
2.23.7	nb_wakeup . . . . .	64
2.23.8	nb_poweroff . . . . .	64
2.23.9	nb_reset_factory . . . . .	64
2.23.10	nb_reset_statistics . . . . .	64
2.23.11	nb_wlan_tracking . . . . .	65
2.23.12	nb_wanlink_activate . . . . .	65
2.23.13	nb_wanlink_deactivate . . . . .	65
2.23.14	nb_wanlink_priorize . . . . .	65
2.23.15	nb_wanlink_weight . . . . .	66
2.23.16	nb_reset_debug_level . . . . .	66
2.23.17	nb_set_debug_level . . . . .	66
2.23.18	nb_get_debug_level . . . . .	66



# 1 Introduction

This manual describes the SDK API extensions to the standard library of version #SDK\_VERSION### They 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
- Bluetooth functions
- Various network-related functions
- OPC-UA 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 written 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
-----	-------------

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.

Available media devices:

usb0	first USB device
storage0	first extended storage device

### 2.2.2 nb\_media\_umount

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

The nb\_media\_umount function unmounts the specified media device.

dev	device name (e.g usb0, storage0)
-----	----------------------------------

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.

<code>fd</code>	file descriptor
<code>type</code>	can be either <code>MODBUS_TYPE_TCP</code> or <code>MODBUS_TYPE_RTU</code>

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.

<code>fd</code>	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 or communicating via Modbus-TCP - RTU Gateway.

<code>fd</code>	file descriptor
<code>slave</code>	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

```
void 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

### 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_replay_raw_response` function will reply to a modbus request.

<code>fd</code>	file descriptor
<code>response</code>	the raw modbus response

The `nb_modbus_replay_raw_response` function will return the number of bytes sent on success. 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.

<code>fd</code>	file descriptor
<code>request</code>	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.

<code>fd</code>	file descriptor
<code>addr</code>	address of bits to read
<code>len</code>	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_input_bits`

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

The `nb_modbus_read_input_bits` function reads the input bits from the remote device.

<code>fd</code>	file descriptor
<code>addr</code>	address of bits to read
<code>len</code>	length of data to read

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

### 2.3.12 `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.

<code>fd</code>	file descriptor
<code>addr</code>	address of registers to read
<code>len</code>	length of data to read

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

### 2.3.13 `nb_modbus_read_input_regs`

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

The `nb_modbus_read_input_regs` function reads the input registers from the remote device.

<code>fd</code>	file descriptor
<code>addr</code>	address of registers to read
<code>len</code>	length of data to read

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

### 2.3.14 `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.

<code>fd</code>	file descriptor
<code>addr</code>	address of bits to write
<code>length</code>	length of array
<code>data</code>	array to write

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

### 2.3.15 `nb_modbus_write_input_bits`

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

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

<code>fd</code>	file descriptor
<code>addr</code>	address of input bits to write
<code>length</code>	length of array
<code>data</code>	array to write

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

### 2.3.16 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.

### 2.3.17 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.

### 2.3.18 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.

### 2.3.19 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.

<code>fd</code>	file descriptor
<code>req</code>	request
<code>resp</code>	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
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 specified in the struct msg which includes the following fields:

number	recipient's phone number
report	request delivery report (if set to yes)
gateway	the SMS gateway used for sending the message
sim	the SIM over which the message shall be sent
modem	the modem used for sending (deprecated)
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)
int nb_email_send (string rcpt, string subj, string msg, string att)
```

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
att	an optional file to be sent as attachment

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

The following functions can be used to send or manage E-Mails on a remote server.

Supported protocols are:

smtp	SMTP
smtps	SMTP over SSL
imap	IMAP
imaps	IMAP over SSL
pop3	POP3
pop3s	POP3 over SSL

### 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

```
struct 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\_done

```
int nb_config_done (void)
```

The nb\_config\_done function can be used to check if all modify scripts have completed after a config change.

Returns 0 on ready, 1 on busy and -1 on error.

### 2.7.4 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 USSD Queries

### 2.10.1 nb\_ussd\_query

```
string nb_ussd_query (string modem, string msg)
```

The `nb_ussd_query` function can be used to send Unstructured Supplementary Service Data messages to a particular modem. A typical USSD message starts with an asterisk (\*) followed by digits that comprise commands or data. Groups of digits may be separated by additional asterisks. The message is terminated with a number sign (#).

`modem` the modem to query (e.g. Mobile1)  
`query` the USSD message (e.g. \*135#)

Returns a string holding the response of the USSD query.

## 2.11 File Transfers

The file transfer functions can be used to transfer files from or to a remote server using an URL according to RFC 3986.

Supported protocols are:

http	HTTP
https	HTTP over SSL
ftp	FTP
ftps	FTP over SSL
tftp	TFTP
sftp	SSH FTP

Please note that all functions operate on files in the SDK sandbox (which is /mnt/sdk on the host system).

### 2.11.1 nb\_transfer\_get

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

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
header	custom HTTP headers

Returns -1 on error.

Custom HTTP headers have to be provided as follows:

```
<header1>\n<header2>\n<header3>\n...<headerN>
```

### 2.11.2 nb\_transfer\_put

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

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 should be sent
header	custom HTTP headers

Returns -1 on error.

Custom HTTP headers have to be provided as follows:

```
<header1>\n<header2>\n<header3>\n...<headerN>
```

### 2.11.3 nb\_transfer\_post

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

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 should be sent
pp	additional POST parameters
resp	the path to response file
header	custom HTTP headers

Returns -1 on error.

POST parameters have to be provided as follows:

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

If provided, the server's response will be stored in the resp file.

Custom HTTP headers have to be provided as follows:

```
<header1>\n<header2>\n<header3>\n...<headerN>
```

### 2.11.4 nb\_transfer\_list

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

The nb\_transfer\_list function can be used to retrieve the list of files 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 directory to be listed
header	custom FTP commands (see RFC959)

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

Custom FTP commands have to be provided as follows:

```
<command1>\n<command2>\n<command3>\n...<commandN>
```

### 2.11.5 nb\_transfer\_delete

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

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
header	custom FTP commands (see RFC959)

Returns -1 on error.

Custom FTP commands have to be provided as follows:

```
<command1>\n<command2>\n<command3>\n...<commandN>
```

## 2.12 LED

### 2.12.1 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.

### 2.12.2 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.

### 2.12.3 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
```

## 2.13 Config/Software Update

The following functions can be used to trigger a configuration or software update of the system. An Uniform Resource Locator (URL) can have the following format:

```
http://<username>:<password>@<host>:<port>/<path>
https://<username>:<password>@<host>:<port>/<path>
ftp://<username>:<password>@<host>:<port>/<path>
sftp://<username>:<password>@<host>:<port>/<path>
tftp://<host>/<path>
file:///<path>
```

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.

### 2.13.1 `nb_update_status`

```
string nb_update_status (void)
```

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

The following strings can be returned:

no update is running

```
xy update has started
xy update is running
xy update has succeeded
xy update has failed
xy is up-to-date
```

With xy being one of:

```
config
software
license
firmware
sshkeys
```

### 2.13.2 nb\_update\_config

```
int nb_update_config (string url)
int nb_update_config (string url, bool incremental)
```

The `nb_update_config` function will perform a configuration update from the specified URL. If `incremental` is `false` configuration parameters which are not included in the new configuration will be reset to factory defaults. Otherwise they will be omitted.

<code>url</code>	the URL of the config file
<code>incremental</code>	perform incremental update

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.

### 2.13.3 nb\_update\_software

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

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

<code>url</code>	the URL of the software image
------------------	-------------------------------

Returns zero on success.

### 2.13.4 nb\_update\_sshkeys

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

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

<code>url</code>	the URL of the keys file
------------------	--------------------------

Returns zero on success.

## 2.14 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.

### 2.14.1 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 registers 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 struct which can be used for other page functions.

### 2.14.2 nb\_userpage\_register

Registers a new page which is also accessible by non-admin users, see nb\_page\_register().

### 2.14.3 nb\_page\_unregister

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

The nb\_page\_unregister function can be used to unregister a page again.

page	page struct
------	-------------

Returns -1 on error, otherwise zero.

### 2.14.4 nb\_page\_request

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

The nb\_page\_request function listens for incoming requests.

page	page struct
------	-------------

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

### 2.14.5 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 struct
fmt	format string

Returns -1 on error and zero on success.

### 2.14.6 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 struct
------	-------------

Returns -1 on error and zero on success.

## 2.15 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 structs 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 structs 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
nil://Nil1    Null device
```

The following states are possible:

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

### 2.15.1 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 struct 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:

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

### 2.15.2 `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 structs.

### 2.15.3 `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 struct.

#### 2.15.4 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 struct.

#### 2.15.5 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 struct.

#### 2.15.6 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.

#### 2.15.7 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.

### 2.15.8 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.

### 2.15.9 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.

### 2.15.10 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.

## 2.16 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.

### 2.16.1 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.

### 2.16.2 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.

### 2.16.3 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.

### 2.16.4 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

### 2.16.5 nb\_snmp\_unlink

```
int nb_snmp_unlink (void)
```

The nb\_snmp\_unlink function disconnects any MIB entities from the agent. Returns -1 on error.

### 2.16.6 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)
int nb_snmp_host (string host, int port, int version, string user,
                 string password, string auth, string priv, string engine)
```

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)
engine	engine ID

Returns -1 on error.

### 2.16.7 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 has to be set with nb\_snmp\_host prior to using that function.

oid	the queried OID
-----	-----------------

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

### 2.16.8 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 has to be set with nb\_snmp\_host 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.

### 2.16.9 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)
```

```
int nb_snmp_trap (string host, int port, int version, string user,
                 string password, string auth, string priv, string engine)
```

The nb\_snmp\_trap function will set the host for sending SNMP traps. The same parameters as for nb\_snmp\_host apply. Returns -1 on error.

### 2.16.10 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 ('n' for null, 'i' for integer and 's' for octet string)
value	value to be sent

Please note that a traphost has to be set with nb\_snmp\_traphost prior to using this function.

Returns -1 on error.

### 2.16.11 nb\_snmp\_inform

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

The nb\_send\_inform function will send an SNMPv3 inform 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 with nb\_snmp\_traphost prior to using this function.

Returns -1 on error.

## 2.17 MQTT

### 2.17.1 nb\_mqttlib\_new

```
int nb_mqttlib_new(string id, bool clean_session)
```

id	String to use as the client id. If NULL, a random client id will be generated. If id is NULL, clean_session must be true.
clean_session	set to true to instruct the broker to clean all messages and subscriptions on disconnect, false to instruct it to keep them.

Creates a mqtt handle. Use the returned handle in all other mqtt functions.

Returns a mqtt handle (mh). On failure:

-22	(ENOMEM) on out of memory.
-12	(EINVAL) on invalid input parameters.

### 2.17.2 nb\_mqttlib\_free

```
int nb_mqttlib_free(int mh)
```

mh	An mqtt handle, generated with nb_mqttlib_new().
----	--

Return values:

-22	(ENOMEM) on out of memory.
-12	(EINVAL) on invalid input parameters.

### 2.17.3 nb\_mqttlib\_set\_protocol\_version

```
int nb_mqttlib_set_protocol_version (int mh, string protocol_version)
```

mh	The connection to be used.
protocol_version	The MQTT protocol version. (allowed values: "V31", "V311", "V5". defaults to: "V31")

Call this function before calling nb\_mqttlib\_connect().

Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

### 2.17.4 nb\_mqttlib\_set\_tls



```
int nb_mqttlib_set_tls (int mh, string cacert, string clientcert, string
pkey)
```

mh	The connection to be used.
cacert	The certificate to be used.
clientcert	(Optional) PEM encoded certificate file for this client. If empty, pkey must also be empty and no client certificate will be used.
pkey	(Optional) Private key for this device (password encrypted key not supported). If empty, clientcert must also be empty and no client certificate will be used.

Call this function before calling nb\_mqttlib\_connect().

Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

### 2.17.5 nb\_mqttlib\_set\_tls\_insecure

```
int nb_mqttlib_set_tls_insecure(int mh, bool value)
```

mh	The connection to be used.
value	if set to false, the default, certificate hostname checking is performed. If set to true, no hostname checking is performed and the connection is insecure.

Call this function before calling nb\_mqttlib\_connect().

Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

### 2.17.6 nb\_mqttlib\_set\_user\_pw

```
int nb_mqttlib_set_user_pw (int mh, string user, string password)
```

mh	The connection to be used.
user	The username to be used.
password	The password to be used.

Call this function before calling nb\_mqttlib\_connect().

Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

### 2.17.7 nb\_mqttlib\_get\_callback\_connect

```
struct nb_mqttlib_get_callback_connect(int mh, int timeout, array
    property_ids)
```

md            mosquitto descriptor.  
 timeout      timeout in ms.

MQTT version 5 (optional):

property\_ids   array of property ids as integers. Only properties with those ids are returned (check mqttlib-example.are for property numbers).

Always check the return code with this function after a connect call.

return value is a struct containing:

rc:            int  
               0 - success  
               1 - connection refused (unacceptable protocol version)  
               2 - connection refused (identifier rejected)  
               3 - connection refused (broker unavailable)  
               and MQTT version 5 return codes  
 props:       array (size 0-\*) of array (size 2-3) [ int property\_id, ? property\_value1, ? property\_value2 ]

### 2.17.8 nb\_mqttlib\_get\_callback\_message

```
struct nb_mqttlib_get_callback_message(int mh, int timeout, array
    property_ids)
```

md            mosquitto descriptor.  
 timeout      timeout in ms.

MQTT version 5 (optional):

property\_ids   array of property ids as integers. Only properties with those ids are returned (check mqttlib-example.are for property numbers).

return value is a struct containing:

```
retain:      int
qos:         int
mid:         int
message:     struct {topic:string, message:string}
props:       array (size 0-*) of array (size 2-3) [ int property_id, ? prop-
              erty_value1, ? property_value2 ]
```

### 2.17.9 nb\_mqttlib\_get\_callback\_subscribe

```
struct nb_mqttlib_get_callback_subscribe(mh, timeout, array property_ids)
```

```
md          mosquitto descriptor.
timeout     timeout in ms.
property_ids array of property ids as integers. Only properties with those ids
            are returned (check mqttlib-example.are for property numbers).
```

return value is a struct:

```
mid:        int
qos_count:  int
granted_qos: array
props:      array
```

```
On timeout  Empty struct.
```

### 2.17.10 nb\_mqttlib\_get\_callback\_unsubscribe

```
struct nb_mqttlib_get_callback_unsubscribe(int mh, int timeout, array
      property_ids)
```

```
md          mosquitto descriptor.
timeout     timeout in ms.
```

MQTT version 5 (optional):

```
property_ids array of property ids as integers. Only properties with those ids
            are returned (check mqttlib-example.are for property numbers).
```

return value is a struct containing:

```
mid:        int
props:      array (size 0-*) of array (size 2-3) [ int property_id, ? prop-
              erty_value1, ? property_value2 ]
```

### 2.17.11 nb\_mqttlib\_connect

```
int nb_mqttlib_connect (int mh, string host, int port, int keepalive,
    array properties)
```

mh	The connection to be used.
host	IP-address or hostname of the broker to connect to.
port	network port to connect to (default value used by mqtt is 1883).
keepalive	The mqtt keepalive time (in seconds, recommended value ~60).

MQTT version 5 (optional):

properties	MQTT v5 properties to send.
------------	-----------------------------

array of property arrays. The array can be created the following way:

property1	= mkarray(MQTT_PROP_SESSION_EXPIRY_INTERVAL, 30);
property2	= mkarray(MQTT_PROP_MAXIMUM_PACKET_SIZE, 4096);
properties	= mkarray(property1, property2);

Call this function first before calling the other MQTT functions. To e.g. subscribe/publish on a topic on the broker you connect here, use the same "mh" handler when calling the corresponding MQTT function.

Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

MQTT version 5:

2	(MOSQ_ERR_PROTOCOL) if any property is invalid for use with PUBLISH.
22	(MOSQ_ERR_DUPLICATE_PROPERTY) if a property is duplicated where it is forbidden.

### 2.17.12 nb\_mqttlib\_disconnect

```
int nb_mqttlib_disconnect (int mh, int reason_code, array properties)
```

mh	connection to be used.
----	------------------------

MQTT version 5:

reason_code	the disconnect reason code.
properties	(Optional) MQTT v5 properties to send.

Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

MQTT version 5:

2	(MOSQ_ERR_PROTOCOL) if any property is invalid for use with PUBLISH.
22	(MOSQ_ERR_DUPLICATE_PROPERTY) if a property is duplicated where it is forbidden.

### 2.17.13 nb\_mqttlib\_publish

```
int nb_mqttlib_publish (int mh, string topic, string message, int qos,
    int retain, array properties)
```

The nb\_mqttlib\_publish function can be used to publish a mqtt message on a specified topic.

mh	The connection to be used.
topic	the topic the message will be published to topics are case sensitive and must consist of at least one character / is used as topic level separator topics shouldn't start with / because it wastes one layer + and # are wildcard characters and are not valid in publish topics \$ is a character reserved for mqtt system messages and can not be used
qos	quality of service used for the message 0 - publish at most one (fire and forget) 1 - publish at least once (message can also be delivered more than once) 2 - publish exactly once (guarantees that each message is received only once) higher qos levels result in more traffic
retain	is used to make the broker store the last message sent to a specific topic 0 - the message will not be stored 1 - the message will be stored and send to a new subscriber of that topic
message	the message that is going to be published

MQTT version 5 (optional):

properties	MQTT v5 properties to send.
------------	-----------------------------

Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

MQTT version 5:

2	(MOSQ_ERR_PROTOCOL) if any property is invalid for use with PUBLISH.
22	(MOSQ_ERR_DUPLICATE_PROPERTY) if a property is duplicated where it is forbidden.

### 2.17.14 nb\_mqttlib\_set\_will

```
int nb_mqttlib_set_will(int mh, string topic, string message, int qos,
    int retain, array properties)
```

Call this function before calling nb\_mqttlib\_connect(). Same parameters and return values as nb\_mqttlib\_publish().

### 2.17.15 nb\_mqttlib\_clear\_will

```
int nb_mqttlib_clear_will(int mh)
```

mh            The connection to be used.

Remove a previously configured will. Call this function before calling nb\_mqttlib\_connect(). Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

### 2.17.16 nb\_mqtt\_publish

This function is deprecated and should no longer be used. It will be removed in future releases.

```
int nb_mqtt_publish (string host, int port, string username, string
    password, string topic, int qos, int retain, string message)
```

The nb\_mqtt\_publish function can be used to publish a mqtt message on a specified

topic.

host	IP-address or hostname of the broker to connect to
port	network port to connect to (default value is 1883)
username	necessary if the mqtt broker requires authentication
password	optional password for the username mentioned above
topic	the topic the message will be published to topics are case sensitive and must consist of at least one character / is used as topic level separator topics shouldn't start with / because it wastes one layer + and # are wildcard characters and are not valid in publish topics \$ is a character reserved for mqtt system messages and can not be used
qos	quality of service used for the message 0 = publish at most one (fire and forget) 1 = publish at least once (message can also be delivered more than once) 2 = publish exactly once (guarantees that each message is received only once) higher qos levels result in more traffic
retain	is used to make the broker store the last message sent to a specific topic 0 = the message will not be stored 1 = the message will be stored and send to a new subscriber of that topic
message	the message that is going to be published

Returns -1 on error, otherwise zero.

### 2.17.17 nb\_mqttlib\_subscribe

```
int nb_mqttlib_subscribe (int mh, string topic, int qos, int options,
    array properties)
```

The nb\_mqttlib\_subscribe function can be used to subscribe to a specified topic.

mh	The connection to be used.
topic	the topic the message will be published to topics are case sensitive and must consist of at least one character / is used as topic level separator topics shouldn't start with / because it wastes one layer + and # are wildcard characters and are not valid in publish topics \$ is a character reserved for mqtt system messages and can not be used
qos	quality of service used for the message 0 - publish at most one (fire and forget) 1 - publish at least once (message can also be delivered more than once) 2 - publish exactly once (guarantees that each message is received only once) higher qos levels result in more traffic

MQTT version 5 (optional):

options	options to apply to this subscription, OR'd together. Set to 0 to use the default options, otherwise choose from the list: MQTT_SUB_OPT_NO_LOCAL: with this option set, if this client publishes to a topic to which it is subscribed, the broker will not publish the message back to the client. MQTT_SUB_OPT_RETAIN_AS_PUBLISHED: with this option set, messages published for this subscription will keep the retain flag as was set by the publishing client. The default behaviour without this option set has the retain flag indicating whether a message is fresh/stale. MQTT_SUB_OPT_SEND_RETAIN_ALWAYS: with this option set, pre-existing retained messages are sent as soon as the subscription is made, even if the subscription already exists. This is the default behaviour, so it is not necessary to set this option. MQTT_SUB_OPT_SEND_RETAIN_NEW: with this option set, pre-existing retained messages for this subscription will be sent when the subscription is made, but only if the subscription does not already exist. MQTT_SUB_OPT_SEND_RETAIN_NEVER: with this option set, pre-existing retained messages will never be sent for this subscription.
properties	MQTT v5 properties to send.

Return values (check mqttlib-example.are for all return values):



0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

MQTT version 5:

2	(MOSQ_ERR_PROTOCOL) if any property is invalid for use with PUBLISH.
22	(MOSQ_ERR_DUPLICATE_PROPERTY) if a property is duplicated where it is forbidden.

### 2.17.18 nb\_mqttlib\_unsubscribe

```
int nb_mqttlib_unsubscribe (int mh, string topic, array properties)
```

mh	The connection to be used.
topic	the unsubscription pattern. See mqttlib_subscribe for pattern description.

MQTT version 5:

properties	(Optional) MQTT v5 properties to send.
------------	--

Return values (check mqttlib-example.are for all return values):

0	(MOSQ_ERR_SUCCESS) on success
1	(MOSQ_ERR_NOMEM) if an out of memory condition occurred.
3	(MOSQ_ERR_INVALID) if the input parameters were invalid

MQTT version 5:

2	(MOSQ_ERR_PROTOCOL) if any property is invalid for use with PUBLISH.
22	(MOSQ_ERR_DUPLICATE_PROPERTY) if a property is duplicated where it is forbidden.

## 2.18 CAN

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

### 2.18.1 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.

### 2.18.2 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.

### 2.18.3 nb\_can\_close

```
int nb_can_close (int socket)
```

socket	socket descriptor
--------	-------------------

The nb\_can\_close function closes the specified socket descriptor and disables the associated interface. Returns -1 on error.

### 2.18.4 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).

### 2.18.5 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.

### 2.18.6 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 bytes)
len	size of received data

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

### 2.18.7 nb\_can\_sendmsg

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

socket	socket descriptor
msg	message struct (id + data)

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

## 2.19 Bluetooth

Bluetooth devices can be configured for scanning or advertising (and also can be disabled) by calling the `nb_bt_mode` function. Discovered devices can be retrieved by using the `nb_bt_dev_list` and additional information about devices can be requested by the `nb_bt_dev_info` function.

Adapter properties (ada stands for adapter) can be accessed by the functions `nb_bt_ada_property_getall`, which is used to retrieve all properties for a single adapter and `nb_bt_ada_property_set`, which is used to modify a single adapter property.

Advertising (adv stands for advertising) properties can be accessed by the functions `nb_bt_adv_property_getall`, which is used to retrieve all advertising properties for a single adapter and `nb_bt_adv_property_set`, which is used to modify a single advertising property.

### 2.19.1 nb\_bt\_dev\_list

```
struct nb_bt_dev_list (int adapter)
```

The `nb_bt_dev_list` function is used in scan mode and returns a list of devices (mac as string).

<code>adapter</code>	the adapter to use (see below)
----------------------	--------------------------------

Returns a array of device string

The adapter can a a numerical index or one of the predefined constants:

<code>BT1</code>	The first adapter
<code>BT2</code>	The second adapter

### 2.19.2 nb\_bt\_dev\_info

```
struct nb_bt_dev_info (int adapter, string device)
```

The `nb_bt_dev_info` function is used in scan mode and returns the device info of an existing device in the devices list (see see `nb_bt_dev_list`)

<code>adapter</code>	the adapter to use (see below)
<code>device</code>	device mac separated by colons e.g. "00:11:22:33:44:55"

Returns the device info of the referenced adapter as a struct with the following properties (some may be optional):

```

string Address (e.g. "00:11:22:33:44:55")
string Addresstype
string Name
string Alias
int Class
int Appearance
string Icon
boolean Paired
boolean Trusted
boolean Blocked
boolean LegacyPairing
int RSSI
boolean Connected
array of string UUIDs
string Modalias
int Adapter (e.g. BT1 or BT2)
array of struct { int Key, array of int Value } ManufacturerData
array of struct { string Key, array of int Value } ServiceData
int TxPower
boolean ServicesResolved
array of int AdvertisingFlags
array of struct { int Key, array of int Value } AdvertisingData
    
```

The adapter can be a numerical index or one of the predefined constants:

```

BT1    The first adapter
BT2    The second adapter
    
```

### 2.19.3 nb\_bt\_dev\_del

```

int nb_bt_dev_del (int adapter, string device)
    
```

The nb\_bt\_dev\_del function is used in scan mode to remove manually a bluetooth device from the devices list

```

adapter    the adapter to use (see below)
device     device mac (separated by colons e.g. "00:11:22:33:44:55")
    
```

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

The adapter can be a numerical index or one of the predefined constants:

```

BT1    The first adapter
    
```

BT2 The second adapter

### 2.19.4 nb\_bt\_ada\_property\_getall

```
struct nb_bt_ada_property_getall (int adapter)
```

The nb\_bt\_ada\_property\_getall function returns all adapter properties

adapter      the adapter to use (see below)

Returns all adapter properties as a struct with the following properties (some may be optional):

```
string Address (e.g. "00:11:22:33:44:55")
string Addresstype
string Name
string Alias
int Class
boolean Powered
boolean Discoverable
int DiscoverableTimeout
boolean Pairable
int PairableTimeout
boolean Discovering
array of string UUIDs
string Modalias
```

The adapter can a a numerical index or one of the predefined constants:

BT1 The first adapter  
BT2 The second adapter

### 2.19.5 nb\_bt\_ada\_property\_set

```
int nb_bt_ada_property_set (int adapter, int type, value)
```

The nb\_bt\_ada\_property\_set function is used to set an adapter property

adapter      the adapter to use (see below)  
type          the property name (see below)  
value          the property value to set. The type depends on the type parameter.

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

The adapter can be a numerical index or one of the predefined constants:

```
BT1    The first adapter
BT2    The second adapter
```

Currently supported adapter property names are

```
BT_ADA_ALIAS    sets the bluetooth friendly name (string).
BT_ADA_POWERED switches the adapter power on or off (bool).
```

### 2.19.6 nb\_bt\_adv\_property\_getall

```
struct nb_bt_adv_property_getall (int adapter)
```

The `nb_bt_adv_property_getall` function returns all advertise properties

```
adapter    the adapter to use (see below)
```

Returns all advertise properties as a struct with the following properties (some may be optional):

```
string Type
array of string ServiceUUIDs
array of string SolicitUUIDs
array of struct { int Key, array of int Value } ManufacturerData
array of struct { string Key, array of int Value } ServiceData
array of struct { int Key, array of int Value } Data
boolean Discoverable
int DiscoverableTimeout
string LocalName
array of string Includes
int Appearance
int Duration
int Timeout
```

The adapter can be a numerical index or one of the predefined constants:

```
BT1    The first adapter
BT2    The second adapter
```

### 2.19.7 nb\_bt\_adv\_property\_set

```
int nb_bt_adv_property_set (int adapter, int type, value)
```

The `nb_bt_adv_property_set` function is used to set an advertise property

<code>adapter</code>	the adapter to use (see below)
<code>type</code>	the property name (see below)
<code>value</code>	the property value to set. The type depends on the type parameter.

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

The adapter can a a numerical index or one of the predefined constants:

<code>BT1</code>	The first adapter
<code>BT2</code>	The second adapter

Currently supported advertisement property names are:

<code>BT_ADV_LOCAL_NAME</code>	sets the local name (type string)
--------------------------------	-----------------------------------

### 2.19.8 `nb_bt_mode`

```
int nb_bt_mode (int adapter, int mode)
```

Then `nb_bt_mode` function is used to start scanning devices or start advertising or stop all.

<code>adapter</code>	the adapter to use (see below)
<code>mode</code>	the mode to set (see below)

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

The adapter can a a numerical index or one of the predefined constants:

<code>BT1</code>	The first adapter
<code>BT2</code>	The second adapter

The mode can be one of the following:

<code>BT_MODE_OFF</code>	disables the device.
<code>BT_MODE_SCAN</code>	sets the device in scan-mode.
<code>BT_MODE_ADV</code>	sets the device in advertising moude.



## 2.20 Network

### 2.20.1 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.

### 2.20.2 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.

### 2.20.3 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.

### 2.20.4 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.

### 2.20.5 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.

### 2.20.6 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.

## 2.21 OPC-UA

The OPC-UA functions below offer facilities to

- connect to an OPC-UA server
- browse the node store
- search for nodes
- read/write values from/to nodes

Please note that only integer, string, double and boolean values are currently supported.

A node struct is usually represented as struct with the following fields:

namespace-index the namespace index node-id the node identifier browse-name the name shown when browsing display-name the display name

### 2.21.1 nb\_opcua\_connect

```
int nb_opcua_connect (string url)
```

The nb\_opcua\_connect function will connect to an OPC-UA server.

url	the server URL ("opc.tcp:// . . . : . . . ")
-----	--

Returns a client session descriptor or -1 on error.

### 2.21.2 nb\_opcua\_browse

```
int nb_opcua_browse (int client, int nindex, value nid, int depth)
```

The nb\_opcua\_browse function will browse recursively the children of the specified node.

client	the descriptor of the client session
nindex	the namespace index of the node
nid	the node identifier from where browsing starts
depth	specifies how deep the node tree will be descended

Returns an array of node structs or void on error.

### 2.21.3 nb\_opcua\_search

```
int nb_opcua_search (int client, value pattern)
```

The nb\_opcua\_search function will search in the entire namespace at the server for nodes matching the given pattern.

client	the descriptor of the client session
pattern	the pattern to search for

Returns an array of node structs with found nodes or void on error.

### 2.21.4 nb\_opcua\_read

```
value nb_opcua_read (int client, int nindex, int nid)
```

The nb\_opcua\_read function will read the value of the given node.

client	the descriptor of the client session
nindex	the namespace index of the node
nid	the node identifier

Returns the value of the given node or void on error.

### 2.21.5 nb\_opcua\_write

```
int nb_opcua_write (int client, int nindex, int nid, value val)
```

The nb\_opcua\_write function will change the value of the given node.

client	the descriptor of the client session
nindex	the namespace index of the node
nid	the node identifier
val	the new value for the node

Returns zero or -1 on error.

### 2.21.6 nb\_opcua\_disconnect

```
int nb_opcua_disconnect (int client)
```

The nb\_opcua\_disconnect function will disconnect the client from the server

client	the descriptor of the client session
--------	--------------------------------------

Returns -1 on error.

## 2.22 Certificates

### 2.22.1 nb\_cert\_install

```
int nb_cert_install (string cert, string password, string url)
```

The nb\_cert\_install function installs a certificate from the given URL.

Valid certificate identifiers are:

```
- root
- webserver
- sshd
- mqttbroker
- openvpn-tunnel1, openvpn-tunnel2, ...
- openvpn-tunnel1-client0, ...
- wlan1, ...
```

```
- wlan1-1-client1, ...
- ipsectunnell
- other
```

cert	Certificate to be installed
password	Password to decrypt the certificate file (Empty-String if no password is needed)
url	URL to read from (see Config/Software Update for examples of valid URL strings)

Returns -1 on error.

### 2.22.2 nb\_cert\_create

```
int nb_cert_create (string cert)
```

The nb\_cert\_create function creates a certificate and installs it to the system. For valid certificate identifiers look at nb\_cert\_install.

cert	Certificate to be installed
------	-----------------------------

Returns -1 on error.

### 2.22.3 nb\_cert\_enroll

```
int nb_cert_enroll (string cert)
```

The nb\_cert\_enroll function installs a certificate from a SCEP server. For valid certificate identifiers look at nb\_cert\_install. The SCEP context has to be configured to use this function.

cert	Certificate to be installed
------	-----------------------------

Returns -1 on error.

### 2.22.4 nb\_cert\_erase

```
int nb_cert_erase (string cert)
```

The nb\_cert\_erase function removes an installed certificate from the system. For valid certificate identifiers look at nb\_cert\_install.

cert	Certificate to be removed
------	---------------------------

Returns -1 on error.

### 2.22.5 nb\_cert\_read

```
string nb_cert_read (string cert)
```

The nb\_cert\_read reads an installed certificate from the system. For valid certificate identifiers look at nb\_cert\_install.

cert	Certificate to read
------	---------------------

Returns string holding the certificate as ASCII text. If there is no certificate installed for the requested certificate identifier, an empty string is returned.

## 2.23 Other

### 2.23.1 nb\_syslog

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

The nb\_syslog function creates a message in the system log. The message will show up in log when the monitor log level for sdkhost is set to 6 or higher.

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.

### 2.23.2 nb\_syslog\_p

```
int nb_syslog_p (int loglvl, string fmt, ...)
```

Analog to the nb\_syslog function the nb\_syslog\_p function creates a message in the system log. However, now it is possible to chose the priority of the message.

Please refer to sprintf for more information about the format string and additional arguments.

```

loglvl      log level of the message that will be written to syslog
             0 = emerg
             1 = alert
             2 = crit
             3 = err
             4 = warn
             5 = notice
             6 = info
             7 = debug
msg         message to be written to syslog
    
```

Returns -1 on error.

### 2.23.3 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.

### 2.23.4 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 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"
};
    
```

### 2.23.5 nb\_reboot

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

The nb\_reboot function will trigger a system reboot.

delay            the delay in seconds

### 2.23.6 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.

### 2.23.7 nb\_wakeup

```
int nb_wakeup (int wakeup)
```

The nb\_wakeup function will set the time when the system will reboot after a poweroff.

wakeup            the wakeup time in seconds

Returns -1 on error, otherwise zero.

### 2.23.8 nb\_poweroff

```
int nb_poweroff ()
```

The nb\_poweroff function will power off the router.

### 2.23.9 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.

### 2.23.10 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. WANLINK1)

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

### 2.23.11 nb\_wlan\_tracking

```
struct nb_wlan_tracking (string wlanifc)
```

The `nb_wlan_tracking` function will return all WLAN stations seen on the given WLAN interface.

`wlanifc`      the WLAN interface (e.g. WLAN1)

The `nb_wlan_tracking` function will return all WLAN stations seen on the given WLAN interface or an empty string on error.

### 2.23.12 nb\_wanlink\_activate

```
int nb_wanlink_activate (string wanlink)
```

The `nb_wanlink_activate` function will activate a deactivated WAN link.

`wanlink`      the WAN link to activate (e.g. WANLINK1)

Returns -1 on error, otherwise zero.

### 2.23.13 nb\_wanlink\_deactivate

```
int nb_wanlink_deactivate (string wanlink)
```

The `nb_wanlink_deactivate` function will deactivate an active WAN link.

`wanlink`      the WAN link to deactivate (e.g. WANLINK1)

Returns -1 on error, otherwise zero.

### 2.23.14 nb\_wanlink\_priorize

```
int nb_wanlink_priorize (string wanlink, int prio)
```

The `nb_wanlink_priorize` function can be used to change the priority of a WAN link.

`wanlink`      the WAN link to prioritize (e.g. WANLINK1)

`prio`          the new priority

Returns -1 on error, otherwise zero.

### 2.23.15 nb\_wanlink\_weight

```
int nb_wanlink_weight (string wanlink, int weight)
```

The nb\_wanlink\_weight function can be used to change the weight of a WAN link.

wanlink	the WAN link to prioritize (e.g. WANLINK1)
weight	the new weight

Returns -1 on error, otherwise zero.

### 2.23.16 nb\_reset\_debug\_level

```
int nb_reset_debug_level (string service)
```

The nb\_reset\_debug\_level function will reset the debug level of the given service.

service	the service to reset (e.g. link-manager)
---------	--

Returns -1 on error, otherwise zero.

### 2.23.17 nb\_set\_debug\_level

```
int nb_set_debug_level (string service, int level)
```

The nb\_set\_debug\_level function will set the debug level of the given service.

service	the service to set (e.g. link-manager)
level	the debug level to set (0-7)

Returns -1 on error, otherwise zero.

### 2.23.18 nb\_get\_debug\_level

```
int nb_get_debug_level (string service)
```

The nb\_get\_debug\_level function can be used to get the debug level of the given service.

service	the service to set (e.g. link-manager)
---------	--

Returns the requested level or -1 on error.