relaydroid™

API Documentation

V5.11 (2022.09.07)

Please note:

This documentations is for **relaydroid™** devices with v5.10 firmware or newer. For older versions, use the legacy documentations.

This document is for system integrators and programmers. **relaydroid™** devices contain a built-in webserver with a web user interface suitable for most users. This document shows how you can substitute or extend the built-in functionality.

relaydroid™ devices can be controlled externally from a custom program via HTTP or TCP commands.

The API is disabled by default. To enable the API commands, you must set the *"user#1 (web+API)"* username and password to a non-empty value in the embedded user interface (*"SETTINGS->USERS AND PASSWORDS"* menu).

Please note: Connecting this device to a LAN network needs knowledge about Ethernet network configurations. If your are unfamiliar with setting up Ethernet networks please consult a network specialist!

Contents

1. HTTP GET API

1.a) Get the relay states without switching

API v1 command:
http://*relaydroid_address*/api.cgi?p=*user1_password*

API v2 command:
http://*relaydroid_address*/api2.cgi?p= *user1 _password*

answer:
relay states (see 1.d section)

relaydroid_address: the IP or NetBios name of the device
user1_password: the password of the "user#1 (web+API)" user

1.b) Switch a relay ON or OFF continuously without a time limit and get the relay states

The state is stored in a non-volatile memory and is remembered in case of a system reboot.

API v1 command:
http://*relaydroid_address*/api.cgi?p=*user1 _password*&sw=*oc_num*&v=*com*

API v2 command:
http://*relaydroid_address*/api2.cgi?p=*user1 _password*&sw=*oc_num*&v=*com*

answer:
relay states (see 1.d section)

relaydroid_address: the IP or NetBios name of the device
user1_password: the password of the "user#1 (web+API)" user
oc_num: 1, 2, .. n (number of the OC output to switch)
com: 0: switch OFF, 1: switch ON, 2: switch over (ON->OFF; OFF->ON)

1.c) Switch a relay ON/OFF with a time limit and get the relay states

The state is NOT stored in a non-volatile memory and in case of a system reboot the output will be the last stored state.

API v1 command:
http://*relaydroid_address*/api.cgi?p=*user1_password*&t=*minutes*&sw=*oc_num*&v=*command*
OR
http://*relaydroid_address*/api.cgi?p=*user1_password*&t0=*seconds*&sw=*oc_num*&v= *command*
OR
http://*relaydroid_address*/api.cgi?p=*user1_password*&t1=*milliseconds*&sw=*oc_num*&v= *command*

API v2 command:
http://*relaydroid_address*/api2.cgi?p=*user1_password*&t=*minutes*&sw=*oc_num*&v= *command*
OR
http://*relaydroid_address*/api2.cgi?p=*user1_password*&t0=*seconds*&sw=*oc_num*&v= *command*
OR
http://*relaydroid_address*/api2.cgi?p=*user1_password*&t1=*milliseconds*&sw=*oc_num*&v= *command*

answer:
relay states (see 1.d section)

relaydroid_address: the IP or NetBios name of the device
user1_password: the password of the "user#1 (web+API)" user
t or *t0* or *t1*: time limit in *minutes (t)* or *seconds (t0)* or *milliseconds (t1)* (after the given time the selected OC output will switch over automatically and the state will be stored in non-volatile memory)
oc_num: 1, 2, .. n (number of the OC output to switch)
command: 0: switch OFF, 1: switch ON, 2: switch over (ON->OFF; OFF->ON)

1.d) Relay state format of the answers when using *api.cgi* and *api2.cgi*

API v1 answer (api.cgi):

Sequence of '0'-s and '1'-s, showing which open collector output or digital input is ON (1) or OFF(0).

3 OC out e.g.:

`100` (OC1: ON, OC2-3: OFF)

3 OC out + 3 dig. input e.g.:

`100011` (OC1: ON, OC2-3: OFF, D1: OFF, D2-D3: ON)

API v2 answer (api2.cgi):

RELAY_MAX \r\n
R1_name§*R2_name*§...§*Rmax_name*\r\n
R1_default_time§*R2_defaul_ttime*§...§*Rmax_defaul_ttime*\r\n
R1_state§*R2_state*§...§*Rmax_state*\r\n
INPUT_MAX\r\n
D1_state§D*2_state*§...§D*max_state*

RELAY_MAX: the number of OC outputs
Rx_name: the name of the OC output
Rx_default_time: the default ON time set on the web GUI
Rx_state:
- e.g. ON,0: ON without a time limit
- e.g. ON,-: ON with a time limit of less than 1 seconds
- e.g. ON,100: ON with a time limit of 100 seconds (it will be OFF after 100 seconds)
- e.g. OFF: off

INPUT_MAX: the number of digital inputs
Dx_state: ON (closed) or OFF (open)

e.g.:

`3` - the number of OC outputs is 3
`R01§R02§R03` - the name of the outputs is R01, R02 and R03
`1§1§1` - the default ON time is 1, 1 and 1 sec
`ON,10§ON,0§OFF` - the OC1 output is ON for another 10 seconds (ON,10), the OC2 output is ON without a time limit (ON,0), the OC3 output is OFF
`0` - there are 0 digital input ports
empty line because there are no digital inputs

1.e) GET parameters for serial communication (only for COM versions)

These commands can be used to send serial data through the COM port.

API v1 command:
http://*relaydroid_address*/api.cgi?p=*user1_password*&baud=*baudrate*&com=*comdata*

API v2 command:
http://*relaydroid_address*/api2.cgi?p=*user1_password*&baud=*baudrate*&com=*comdata*

answer:
relay states (see 1.d section)

relaydroid_address: the IP or NetBios name of the device
user1_password: the password of the "user#1 (web+API)" user
baudrate: the baud rate in bps. Valid values are between 1200 and 115200
comdata: these characters will be written to the serial port (TX pin) in 8N1 serial format. You can use the URL encoded format to send special characters (in 2 digit HEX value *%00* format)

The above parameters can be used together with the relay controlling parameters from 1.a)-c). So you can control the relays and send serial data too in one GET request.

e.g.: send "abc 123" characters with 9600 bps. '%20' is the HEX format of the 'space' character
http://192.168.2.201/api.cgi?p=userpass&baud=9600&com=abc%20123

e.g.: switch on OC1 and send "abc\r\n" characters with 19200 bps.' %0D%0A' is the HEX format of the '\r' and '\n' characters
http://192.168.2.201/api.cgi?p=userpass&sw=1&v=1&baud=19200&com=abc%0D%0A

Please note that the whole HTTP GET request cannot be longer than 120 characters. This limits the length of the data that can be sent with one request. If the request is too long you will receive the "414 URI Too Long" error message.

2. TCP/IP API

Usage: open a TCP port and send the messages below.
Default port number to open at the IP address of the device: 80 (the server port can be changed in the "NETWORK" menu settings)
The commands end with a \n (new line) character. After receiving the command, the device answers and closes the connection. The connection must be reopened before sending a new command. The device handles only 2 open connections at a time. Connect to the device only when you want to send the command, do not leave the connection open without sending any characters. The device will close any open connections without a network traffic to free up resources.

2.a) Controlling one relay at a time

Message to send (plain text):

`r[1-n] [0-999999999] [user1_password]\n`
OR
`r[1-n] [0-999999999]- [user1_password]\n`
OR
`r[1-n] - [user1_password]\n`

e.g.: `r1 3000 passw\n` - switch on OC1 for 3 seconds
e.g.: `r1 1000- passw\n` - switch on OC1 for 1 seconds and ask for current state of outputs
e.g.: `r1 - passw\n` - ask for current state of outputs

Answer:

OK

OR

[sequence of '0' and '1'] OK

OR

NO

OK means the command format and the password was correct and the command was executed.
NO means the command format or the password was wrong.
The '0' and '1' characters show which OC output is ON (1) or OFF (0). This character sequence is in the answer only if the request contains the '-' (hyphen) character. Without the '-' the answer is simply 'OK'.

r*n*: *n* is the number of the OC output to switch
0-999999999: time limit in millisecs. After the given time the selected OC output will switch OFF automatically. The '0', '1' and '2' values have a special meaning:

- '0' means: switch OFF immediately.
- '1' means: switch ON without a time limit.
- '2' means: switch to the opposite state (ON->OFF, OFF->ON) without a time limit.
- Other values (bigger than 2) are rounded to 100 millisec intervals and they mean a time limit.

The request can contain the '-' (hyphen) character after the number or as a standalone character. Using the '-' character the answer will contain the state of the OC outputs. Using it with no number you can ask for the OC states without modifying any of them.
user1_password: password of the "user#1 (web+API)" user
\n: the new-line character (ASCII 0x0A)

2.b) Controlling all relays at once

Message to send (plain text):
`rx [sequence of '-' '0' '1' '2'] [user1_password]\n`

'-': not changed
'0': switch OFF
'1': switch ON
'2': switch over (ON->OFF, OFF->ON)

e.g.: `rx 0-1 passw\n` - switch OC1 OFF, OC2 not changed, switch OC3 ON

Answer:

[sequence of '0' and '1'] OK

The '0' and '1' characters show which OC output is ON (1) or OFF (0).

2.c) Using a hash signature in the commands (for increased security)

You can use a hash signature instead of the *user1_password* to authorize the commands.
The device has an internal command counter which increases at every received (and valid) command. Using the command counter, you can sign each command with a different hash signature even if the commands are the same. Using a hash signature, you can be sure that your password is protected even if the network traffic is read by someone, and all message is one-time-only (the messages cannot be re-sent).

Message to send (plain text):

`r[1-n] [0-999999999] +[hash]\n`
OR
`r[1-n] [0-999999999]- +[hash]\n`
OR
`r[1-n] - +[hash]\n`
OR
`rx [sequence of '-' '0' '1' '2'] +[hash]\n`

The message format is the same as in the 2.a)-2.b) sections but instead of the *[user1_password]* you can send a *'+'* character and a *[hash].*
The *[hash]* value is the result of the following formula:

SUBSTR(UPPERCASE(MD5(*'[oc_num]-[switch_time]-[user1_password]-[comm_id]'*)),0,12)

So it is the first 12 uppercased character of an MD5 hash. The following string is used:
'[oc_num]-[switch_time]-[user1_password]-[comm_id]'

[oc_num]: the number of the OC output to switch (if the rn formula is used) or *100* (if the rx formula is used)
[switch_time]: the switching time limit as described in the 2.a) section or 0 if the rx formula is used (see 2.b) section)
[user1_password]: password of the "user#1 (web+API)" user
[comm_id]: the actual value of the command counter (an increasing number)

If the device receives a valid command, the command is executed and the command counter is increased by 1. The answer is the same as in the 2.a)-2.b) sections.
If the device receives a command with an invalid hash signature the command is ignored and the following answer will be sent:
[actual_comm_id] NO
If you do not know the actual command id (e.g. because more than one program sends requests to the device), you can read the actual id from the answer and re-send the command with the updated hash.

Examples:

1.) Switch on OC1 for 1000 millisec. The *user1_password* is 'pass' and the command counter is '89'.
The normal command would be (without hash):
`r1 1000 pass\n`

Calculating the hash:
`SUBSTR(UPPERCASE(MD5('1-1000-pass-89')),0,12) = 2DA8F7C5A90D`
Sending the command with hash signature:
`r1 1000 +2DA8F7C5A90D\n`

2.) Switch all OC outputs at once using the rx command format. Switch off OC1, switch on OC2 and OC3. The *user1_password* is 'pass' and the command counter is '91'.
The normal command would be (without hash):
`rx 011 pass\n`

Calculating the hash:
`SUBSTR(UPPERCASE(MD5('100-0-pass-91')),0,12) = 3B1A504199DA`
Sending the command with hash signature:
`rx 011 +3B1A504199DA\n`

3.) php code example for TCP API:
Please note that the code is simplified (there is no error detection)

```
//switch on OC2 for 5 seconds using hash
$oc = 2;                                            //the number of the OC output
$switch_time = 5000;                                //5000 millisec time limit
$pass = 'pass';                                     //user1 password
$commID = 91;                                       //actual value of the command counter
$md5_hash = md5("$oc-$switch_time-$pass-$commID");  //calculate md5 hash
$hash = substr(strtoupper($md5_hash),0,12);         //get the first 12 chars uppercased
$ip = '192.168.1.190';                              //the IP of the relaydroid
$port = 80;                                         //the port of the relaydroid

$msg = "r$oc $switch_time +$hash\r\n";              //message to send

//create and open TCP socket
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
socket_connect($socket, $ip, $port);
socket_write($socket, $msg, strlen($msg));          //send the message
$answer = socket_read($socket, 100);                //read the answer
socket_close($socket);                              //close the socket
```

2.d) Serial communication with TCP API (only for COM versions)

Message to send (plain text):
`rs [baudrate] [user1_password] [comdata]\n`

baudrate: the baud rate in bps. Valid values are between 1200 and 115200
user1_password: the password of the "user#1 (web+API)" user
comdata: the characters to send in 2 digit HEX format (max 50 chars/100 digits)

e.g.: `rs 9600 passw 616263313233\n`
This will send the 'abc123' characters with 9600bps (8N1).
HEX values: 'a':61, 'b':62, 'c':63, '1':31, '2':32, '3':33

Answer on success:
OK
Answer on error:
NO[code]

Please note that the *comdata* cannot be longer than 100 characters. This limits the length of the data that can be sent with one request. If the request is too long you will receive the "*NOL*" error message.