Template:Networking rutxxx configuration example JSON RPC commands

□

Contents

Some Additional Commands

If the commands found in the guide above did not suffice your needs, this section provides a list of additional ones. The commands presented in this section will be for both Linux and Windows operating systems. They should be used as syntax examples for your own purposes.

WiFi clients list

This command returns a list of devices connected to your WLAN and some additional information about the connection.

Windows:

```
{
    "jsonrpc": "2.0", "id": 1, "method": "call", "params":
    [
        "86fc586fa1471622473434ff0176fd66", "iwinfo", "assoclist",
        {
            "device":"wlan0"
        }
    ]
}
```

Linux:

```
curl -d "{ \"jsonrpc\": \"2.0\", \"id\": 1, \"method\": \"call\", \"params\":
```

```
[ \"86fc586fa1471622473434ff0176fd66\", \"iwinfo\", \"assoclist\",
{\"device\":\"wlan0\"} ] }" http://192.168.1.1/ubus
```

The response should look something like this:


```
{"jsonrpc":"2.0","id":1,"result":[0,{"results":
[{"mac":"E4:02:9B:88:09:AA","signal":-32,"noise":-88,"inactive":10,"rx":
{"rate":1000,"mcs":0,"40mhz":false,"short_gi":false},"tx":
{"rate":72200,"mcs":7,"40mhz":false,"short_gi":true}},
{"mac":"D8:C7:71:47:90:E1","signal":-12,"noise":-88,"inactive":400,"rx":
{"rate":1000,"mcs":0,"40mhz":false,"short_gi":false},"tx":
{"rate":72200,"mcs":7,"40mhz":false,"short_gi":true}}]}]}
```


To obtain these values, the Linux **iwinfo** command and **assoclist** parameter (red) are used. Highlighted in green are the devices connected to the router via WiFi as identified by their MAC addresses. The response information about the connection with the device, such as signal strength, noise, time of inactivity (idle time), rx, tx rate, etc., is highlighted in blue.

WiFi information

This command returns information on your WiFi Access Point.

Windows:

```
{
    "jsonrpc": "2.0", "id": 1, "method": "call", "params":
    [
        "a70ceeba344b6046625d8bcec132796c", "iwinfo", "info",
        {
            "device":"wlan0"
        }
    ]
}
```

Linux:

```
curl -d "{ \"jsonrpc\": \"2.0\", \"id\": 1, \"method\": \"call\", \"params\":
[ \"a70ceeba344b6046625d8bcec132796c\", \"iwinfo\", \"info\",
{\"device\":\"wlan0\"} ] }" http://192.168.1.1/ubus
```

Response:


```
{"jsonrpc":"2.0","id":1,"result":[0,
{"phy":"phy0","ssid":"HAL9000","bssid":"00:1E:42:16:D6:68","country":"00","mo
de":"Master","channel":6,"frequency":2437,"txpower":20,
"quality":22,"quality_max":70,"signal":22,"noise":-61,"bitrate":72200,"encryp
tion":
```

{"enabled":false},"hwmodes":["b","g","n"],"hardware":{"name":"Generic MAC80211"}}]}

As with the clients list command described above, to obtain this information the Linux **iwinfo** command is used, but this time with the **info** parameter (red). The relevant information, such as WiFi SSID, WiFi MAC address, WiFi channel, Encryption type, etc., is highlighted in blue

Manufacturing information

This command returns information about the device's manufacturing details like device's Product Code, Serial Number MAC Address, etc.

Windows:

```
{
    "jsonrpc": "2.0", "id": 1, "method": "call", "params":
    [
        "805725a19ab0fba6c2b44ecf2f952fb9","file", "exec",
        {
            "command":"mnf_info", "params":["name", "sn", "mac"]
        }
    ]
}
```

Linux:

```
curl -d "{ \"jsonrpc\": \"2.0\", \"id\": 1, \"method\": \"call\", \"params\": [ \"805725a19ab0fba6c2b44ecf2f952fb9\",\"file\", \"exec\",{ \"command\":\"mnf_info\", \"params\":[\"name\", \"sn\", \"mac\"] } ] }" http://192.168.1.1/ubus
```

Response:

{"jsonrpc":"2.0","id":1,"result":[0,{"code":0,"stdout":"RUT950HG12C0\n1367435694\n001e4216d666\n"}]}

To obtain the manufacturing information the **mnf_info** (highlighted in red) command is used. In this case a query was sent asking for the device's Product Code (name), Serial Number (sn) and MAC Address (mac) (highlighted in red in the query; returned values highlighted in blue). Using *mnf_info*, you can "ask" the router for any type of manufacturing information. Here is the list of possible *mnf_info* parameters:

- **mac** - returns the router's LAN MAC address
- **maceth** - returns the router's WAN MAC address
- **name** - returns the router's Product Code
- **wps** - returns the router's WPS PIN number
- **sn** - returns the router's Serial number

- **batch** - returns the router's Batch number
- **hwver** - returns the router's Hardware Revision number
- **simpin** - returns the router's SIM card's PIN (as it is specified in the **[[{{{name}}}_Mobile]]** section)
- **blver** - returns the router's Bootloader version

GPS Data

This command returns the device's GPS GPS latitude and longitude.

Windows:

```
{
    "jsonrpc": "2.0", "id": 1, "method": "call", "params":
    [
      "456f77f6b686bf5972daa3a26bee60b0","file", "exec",
         {
           "command":"gpsctl", "params":["-ix"]
         }
    ]
}
```

Linux:

```
curl -d
"{\"jsonrpc\":\"2.0\",\"id\":1,\"method\":\"call\",\"params\":[\"5363304b3ed4
ee0806f101295fc52e93\",\"file\",\"exec\",{\"command\":\"gpsctl\",\"params\":[
\"-ix\"]}]}" http://192.168.1.1/ubus
```

Response:

```
{"jsonrpc":"2.0","id":1,"result":[0,{"code":0,"stdout":"-23.612625\n-46.62635
5\n"}]}
```

The blue part in the code are the Latitude and Longitude.

Firmware number

This command returns the device's Firmware version number.

Windows:

```
{
```

```
    "jsonrpc": "2.0", "id": 1, "method": "call", "params":
    [
        "85ea4cb00398d8387b22d8fa6f75f753", "file", "read",
        {
            "path":"/etc/version"
        }
    ]
}
```

Linux:

```
curl -d "{ \"jsonrpc\": \"2.0\", \"id\": 1, \"method\": \"call\", \"params\":
[ \"85ea4cb00398d8387b22d8fa6f75f753\",\"file\", \"read\",{
\"path\":\"/etc/version\"} ] }" http://192.168.1.1/ubus
```

Response:

```
{"jsonrpc":"2.0","id":1,"result":[0,{"data":"RUT9XX_R_00.05.00.5\n"}]}
```

This command (**file**, **read**, highlighted in red) is an alternative to the Linux **cat** command. All you need is to specify the path (in this case **/etc/version**, highlighted in red) to the file that you wish to read.

Reboot

Windows:

```
{
    "jsonrpc":"2.0","id":1,"method":"call","params":
    [
        "5cd4b143b182c07bc578ae3310d6280e","file","exec",
        {
            "command":"reboot","params":["config"]
        }
    ]
}
```

Linux:

```
curl -d
"{\"jsonrpc\":\"2.0\",\"id\":1,\"method\":\"call\",\"params\":[\"5cd4b143b182
c07bc578ae3310d6280e\",\"file\",\"exec\",{\"command\":\"reboot\",\"params\":[
\"config\"]}]}" http://192.168.1.1/ubus
```

Response:

The success response for this command is an empty message. If the response contains no data, the command was executed successfully.

Set SIM card information

In this last example we'll try to change the mobile connection's MTU and Service mode values.

Windows:

```
{
    "jsonrpc":"2.0", "id":1, "method":"call", "params":
    [
        "558a9b03c940e52f373f8c02498952e3", "uci", "set",
        {
            "config":"simcard", "type":"sim1", "match":
            {
                "service":"auto",
                "mtu":"1500"
            },
            "values":
            {
                "service":"lte-only",
                "mtu":"1476"
            }
        }
    ]
}
```

Linux:

```
curl -d "{\"jsonrpc\":\"2.0\", \"id\":1, \"method\":\"call\",
\"params\":[\"558a9b03c940e52f373f8c02498952e3\", \"uci\", \"set\",
{\"config\":\"simcard\", \"type\":\"sim1\", \"match\":{\"service\":\"auto\",
\"mtu\":\"1500\"}, \"values\":{\"service\":\"lte-only\", \"mtu\":\"1476\"} }
] }" http://192.168.1.1/ubus
```

Response:

```
{"jsonrpc":"2.0","id":1,"result":[0]}
```

The command used is *uci set* (highlighted in red). The config file name is **simcard**, section **sim1**, options **mtu** and **service** (configs, sections and options highlighted in orange). The response shown above is a positive response, but don't forget to execute *uci commit* and *luci-reload* afterwards or else your changes will not take effect.

[[Category:{{{name}}} Configuration Examples]]