Milesight UR Series

HTTP API Documentation

> Firmware version: 3x.3.0.12
>
> All /cgi requests use POST https:///cgi with request header x-session-id: to carry the session credential. Supported API fields vary by product model. Use each field according to the actual supported functions.

Contents

1. Status

1.1 Overview

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the request header `x-session-id: <td>`.

1.1.1 Get Overview

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_status",
  "function": "get",
  "values": [{ "base": "summary" }]
}
```

Response

```
{
  "result": [
    {
      "get": [
        {
          "type": "yruo_status",
          "index": 1,
          "value": {
            "firmware": {
              "model": "YR300",
              "pn": "LOGEU123456",
              "sn": "YR300202401010001",
              "firmware_ver": "3.0.0",
              "hardware_ver": "1.0"
            },
            "system": {
              "local_time": "2024-04-09 10:30:00",
              "uptime": 86400,
              "cpu_load": "15%",
              "cpu_temp": 45,
              "memory_free": 128,
              "memory_total": 256,
              "flash_free": 512,
              "flash_total": 1024,
              "tfcard_state": 2,
              "tfcard_free": 4096,
              "tfcard_total": 8192
            },
            "wan": {
              "wan_exist": 1,
              "cur_link": 0,
```

```
            "status": 1,
            "ip": "192.168.1.100",
            "ipv6": "2001:db8::1",
            "mac": "AA:BB:CC:DD:EE:FF",
            "time": "01:23:45"
          },
          "cell": {
            "cur_link": 1,
            "modem_status": "Connected",
            "network": "4G LTE",
            "signal": 4,
            "cur_sim": "SIM1",
            "ip": "10.0.0.1",
            "ipv6": "2001:db8::2",
            "time": "02:30:15",
            "sim_monthly_traffic": "1.2 GB",
            "sim_monthly_sms": 5
          },
          "wlan": {
            "cur_link": 0,
            "status": 4,
            "mode": 0,
            "ssid": "YR300_AP",
            "connected": 3
          },
          "lan": {
            "ip": "192.168.100.1",
            "ipv6": "fe80::1",
            "connected": 5
          },
          "ups_model_num": "UPS-100",
          "ups_sn": "UPS202401010001",
          "ups_firm_ver": "1.0.0",
          "ups_hard_ver": "1.0",
          "ups_power_status": 2,
          "ups_battery": "85%",
          "ups_battery_temp": 30
        }
      }
    ]
  }
]
}
```


Field Description

`firmware` **object (Device information)**

Field	Type	Description
`model`	string	Device model
`pn`	string	Product number (PN); not returned when absent
`sn`	string	Serial number (SN)

Field	Type	Description
`firmware_ver`	string	Software version
`hardware_ver`	string	Hardware version

`system` object (System status)

Field	Type	Description
`local_time`	string	Device local time
`uptime`	number	Uptime (seconds)
`cpu_load`	string	CPU load
`cpu_temp`	number	CPU temperature (°C)
`memory_free`	number	Available memory (MB)
`memory_total`	number	Total memory (MB)
`flash_free`	number	Available Flash (MB)
`flash_total`	number	Total Flash (MB)
`tfcard_state`	number	TF card state: `-1` not supported, `0` not inserted, `1` abnormal, `2` normal. This field and the following TF card fields are only returned when `tfcard_state !== -1`
`tfcard_free`	number	Available TF card space (MB), only valid when `tfcard_state === 2`
`tfcard_total`	number	Total TF card space (MB), only valid when `tfcard_state === 2`

`wan` object (WAN status)

Returned only when the device has a WAN module.

Field	Type	Description
`wan_exist`	number	Whether a WAN module exists: `1` yes, `0` no
`cur_link`	number	Whether it is the current uplink: `1` yes, `0` no
`status`	number	Connection status: `1` connected; other values mean not connected
`ip`	string	IPv4 address
`ipv6`	string	IPv6 address
`mac`	string	MAC address
`time`	string	Connected duration, format `HH:MM:SS`

`cell` object (Cellular status)

Returned only when the device has a cellular module.

Field	Type	Description
`cur_link`	number	Whether it is the current uplink: `1` yes, `0` no
`modem_status`	string	Module connection status
`network`	string	Network type, e.g. `4G LTE`, `3G WCDMA`
`signal`	number	Signal bars (0–5)
`cur_sim`	string	Currently used SIM card, e.g. `SIM1` / `SIM2`
`ip`	string	IPv4 address
`ipv6`	string	IPv6 address, returned only when the device supports cellular IPv6 and is not the EC200A-L0GEU model
`time`	string	Connected duration, format `HH:MM:SS`
`sim_monthly_traffic`	string	Current SIM card traffic this month
`sim_monthly_sms`	number	Current SIM card SMS count this month

`wlan` object (WLAN status)

Returned only when the device has a WLAN module.

Field	Type	Description
`cur_link`	number	Whether it is the current uplink: `1` yes, `0` no
`status`	number	WLAN status code. `31` / `34` mean disabled; for other values see the status enum
`mode`	number	Operating mode: `0` AP (hotspot), `1` STA (client)
`ssid`	string	SSID; not returned when `status` is disabled
`connected`	number	Number of connected clients, returned only in AP mode (`mode === 0`) and when not disabled

`lan` object (LAN status)

Field	Type	Description
`ip`	string	LAN-side IPv4 address
`ipv6`	string	LAN-side IPv6 address
`connected`	number	Number of connected devices

UPS Fields

The following fields are returned only for 4X series devices (at the top level of `value`).

Field	Type	Description
`ups_model_num`	string	UPS model
`ups_sn`	string	UPS serial number
`ups_firm_ver`	string	UPS firmware version
`ups_hard_ver`	string	UPS hardware version
`ups_power_status`	number	Power status: `0` not connected, `1` internal power (battery), `2` external power
`ups_battery`	string	Battery level
`ups_battery_temp`	number	Battery temperature (°C)

1.2 Cellular

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the request header `x-session-id: <td>`.

Top-level response fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version number
`status`	number	Request execution status: `0` success

1.2.1 Get Cellular Status

Request

```
{
  "id": 7,
  "execute": 1,
  "core": "yruo_status",
  "function": "get",
  "values": [{ "base": "yruo_celluar" }]
```

```
}
```

Response


```
{
  "id": 7,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_cell_status",
          "index": 0,
          "value": {
            "modem": {
              "modem_status": "Ready",
              "model": "EC25",
              "version": "EC25EUXGAR08A17M1G_A0.200.A0.200",
              "cur_sim": "SIM1",
              "signal": "13asu (-87dBm)",
              "band": "4G(B41)",
              "rssi": "-86dBm",
              "rscp": "0dBm",
              "rsrp": "-114dBm",
              "rsrq": "-7dB",
              "sinr": "8dB",
              "signal_quality": "poor",
              "register": "Registered (Home network)",
              "imsi": "460000174582843",
              "iccid": "898600951323F2022668",
              "net_provider": "CHINA MOBILE",
              "net_type": "TDD LTE",
              "plmnid": "46000",
              "lac": "59e7",
              "cellid": "215ad47",
              "imei": "865622071515733"
            },
            "network": {
              "status": "Connected",
              "ip": "10.167.134.48",
              "netmask": "255.255.255.224",
              "gate": "10.167.134.49",
              "dns": "211.143.147.120",
              "sec_dns": "112.5.230.54",
              "time": "0 days, 09:00:35",
              "speed": "RX:     0b, TX:     0b",
              "ipv6": "2409:8934:20dc:e5b:c4f4:5d67:7364:c5a8/64",
              "gatev6": "2409:8934:20dc:e5b:5596:9071:1f7b:a768",
              "dnsv6": "2409:8034:2000:0:0:0:0:4",
              "sec_dnsv6": "2409:8034:2000:0:0:0:0:3"
            },
```

```
            "network1": {
              "status": "Connected",
              "ip": "10.167.134.48",
              "netmask": "255.255.255.224",
              "gate": "10.167.134.49",
              "dns": "211.143.147.120",
              "sec_dns": "112.5.230.54",
              "ipv6": "2409:8934:20dc:e5b:c4f4:5d67:7364:c5a8/64",
              "gatev6": "2409:8934:20dc:e5b:5596:9071:1f7b:a768",
              "dnsv6": "2409:8034:2000:0:0:0:0:4",
              "sec_dnsv6": "2409:8034:2000:0:0:0:0:3",
              "time": "0 days, 09:00:35"
            },
            "network2": {
              "status": "Disabled",
              "ip": "0.0.0.0",
              "netmask": "0.0.0.0",
              "gate": "0.0.0.0",
              "dns": "0.0.0.0",
              "sec_dns": "0.0.0.0",
              "ipv6": "::",
              "gatev6": "::",
              "dnsv6": "::",
              "sec_dnsv6": "::",
              "time": "0 days, 00:00:00"
            },
            "network3": {
              "status": "Disabled",
              "ip": "0.0.0.0",
              "netmask": "0.0.0.0",
              "gate": "0.0.0.0",
              "dns": "0.0.0.0",
              "sec_dns": "0.0.0.0",
              "ipv6": "::",
              "gatev6": "::",
              "dnsv6": "::",
              "sec_dnsv6": "::",
              "time": "0 days, 00:00:00"
            },
            "statistics": {
              "sim1": "RX: 0.3 MiB   TX: 0.7 MiB   ALL: 1.0 MiB",
              "sim1_overflow": 0,
              "sim1_monthly_sms": 0,
              "sim1_sms_overflow": 0,
              "sim1_monthly_traffic": "1.04 MB",
              "sim2": "RX: 0.2 MiB   TX: 0.6 MiB   ALL: 0.8 MiB",
              "sim2_overflow": 0,
              "sim2_monthly_sms": 0,
              "sim2_sms_overflow": 0,
              "sim2_monthly_traffic": "881.10 KB"
            }
          }
        }
      ]
    }
  ]
```

```
}
```

Field Description

`modem` object:

Field	Type	Description
`modem_status`	string	Module status, e.g. `Ready`, `Connected`
`model`	string	Module model
`version`	string	Module firmware version
`cur_sim`	string	Currently used SIM card, `SIM1` / `SIM2`
`signal`	string	Signal strength, format `"<asu>asu (<dBm>dBm)"`
`band`	string	Current band, e.g. `4G(B41)`
`rssi`	string	Received signal strength, with unit, e.g. `"-86dBm"`, used in 2G networks
`rscp`	string	Received signal code power, with unit, e.g. `"0dBm"`, used in 3G networks
`rsrp`	string	Reference signal received power, with unit, e.g. `"-114dBm"`, used in 4G/5G networks
`rsrq`	string	Reference signal received quality, with unit, e.g. `"-7dB"`
`sinr`	string	Signal-to-interference-plus-noise ratio, with unit, e.g. `"8dB"`, used in 4G/5G networks
`signal_quality`	string	Signal quality: `excellent` / `good` / `fair` / `poor`
`register`	string	Registration status, e.g. `"Registered (Home network)"`
`imsi`	string	International Mobile Subscriber Identity
`iccid`	string	SIM card unique identifier
`net_provider`	string	Carrier name
`net_type`	string	Network type, e.g. `TDD LTE`, `3G WCDMA`, `2G GSM`
`plmnid`	string	Public Land Mobile Network identifier
`lac`	string	Location Area Code (hexadecimal)
`cellid`	string	Cell ID (hexadecimal)
`imei`	string	International Mobile Equipment Identity

`network` object (default APN) and `network1` / `network2` / `network3` objects (APN1–3):

Field	Type	Description
`status`	string	Connection status: `Connected` / `Disconnected` / `Disabled`
`ip`	string	IPv4 address, `0.0.0.0` when not connected
`netmask`	string	Netmask
`gate`	string	IPv4 gateway
`dns`	string	Primary DNS
`sec_dns`	string	Secondary DNS
`time`	string	Connected duration, format `"D days, HH:MM:SS"`
`speed`	string	Current speed, format `"RX: <value>, TX: <value>"`; only `network` (default APN) includes this field
`ipv6`	string	IPv6 address, `::` when not connected
`gatev6`	string	IPv6 gateway
`dnsv6`	string	IPv6 primary DNS
`sec_dnsv6`	string	IPv6 secondary DNS

`statistics` object:

Field	Type	Description
`sim1`	string	SIM1 traffic summary for this month, format `"RX: X MiB TX: X MiB ALL: X MiB"`
`sim1_overflow`	number	Whether SIM1 traffic exceeded the limit: `1` exceeded, `0` normal
`sim1_monthly_sms`	number	SIM1 SMS count this month
`sim1_sms_overflow`	number	Whether SIM1 SMS exceeded the limit: `1` exceeded, `0` normal
`sim1_monthly_traffic`	string	SIM1 traffic this month, with unit, e.g. `"1.04 MB"`
`sim2`	string	SIM2 traffic summary for this month, same format as `sim1`
`sim2_overflow`	number	Whether SIM2 traffic exceeded the limit
`sim2_monthly_sms`	number	SIM2 SMS count this month
`sim2_sms_overflow`	number	Whether SIM2 SMS exceeded the limit
`sim2_monthly_traffic`	string	SIM2 traffic this month, with unit, e.g. `"881.10 KB"`

1.2.2 Get Signal History

Supports realtime and historical time range queries. When `day` / `week` data volume is large, results are paginated; use the `has_more` field to determine whether to continue requesting.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_status",
  "function": "get",
  "values": [
    {
      "base": "yruo_signal_history",
      "time_range": "realtime",
      "has_more": 0
    }
  ]
}
```

Parameter	Type	Required	Description
`time_range`	string	Yes	Time range: `realtime` (realtime) / `hour` (last 1 hour) / `day` (last 1 day) / `week` (last 1 week)
`has_more`	number	No	Pagination flag, default `0`; when the previous response `has_more` is `1`, use `1` to continue fetching subsequent data

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "signal_history",
          "index": 1,
          "value": {
            "time_range": "realtime",
            "has_more": 0,
            "data": [
              {
                "timestamp": 1775726483,
                "rssi": -87,
                "rsrp": -115,
```

```
                "sinr": 9
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Field Description

`value` object:

Field	Type	Description
`time_range`	string	Returned time range, consistent with the request
`has_more`	number	`1` means there is more data and another request is needed; `0` means all data has been returned
`data`	array	Array of signal data points

`data[]` data points:

Field	Type	Description
`timestamp`	number	Unix timestamp (seconds)
`rssi`	number	Signal strength (dBm), 2G scenario
`rscp`	number	Received signal code power (dBm), 3G scenario, present only for 3G networks
`rsrp`	number	Reference signal received power (dBm), 4G/5G scenario
`sinr`	number	Signal-to-noise ratio (dB), 4G/5G scenario

> The signal metric fields in a data point are returned according to the current network type; fields for non-current standards may be absent.

1.2.3 Get Traffic History

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_status",
  "function": "get",
  "values": [
    {
      "base": "yruo_traffic_history",
```

```
      "time_range": "day"
    }
  ]
}
```

Parameter	Type	Required	Description
`time_range`	string	Yes	Time range: `day` (today) / `month` (this month) / `year` (this year)

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "traffic_history",
          "index": 0,
          "value": {
            "time_range": "day",
            "sim1_data": [
              { "timestamp": 1775692800, "rx_bytes": 672, "tx_bytes": 8736 },
              { "timestamp": 1775696400, "rx_bytes": 2358, "tx_bytes": 3154 }
            ],
            "sim2_data": [
              { "timestamp": 1775692800, "rx_bytes": 0, "tx_bytes": 0 },
              { "timestamp": 1775696400, "rx_bytes": 0, "tx_bytes": 0 }
            ],
            "dual_data": [
              { "timestamp": 1775692800, "rx_bytes": 672, "tx_bytes": 8736 },
              { "timestamp": 1775696400, "rx_bytes": 2358, "tx_bytes": 3154 }
            ]
          }
        }
      ]
    }
  ]
}
```

Field Description

`value` object:

Field	Type	Description
`time_range`	string	Returned time range, consistent with the request

Field	Type	Description
`sim1_data`	array	SIM1 traffic data point array
`sim2_data`	array	SIM2 traffic data point array
`dual_data`	array	Dual-SIM combined traffic data point array

Data point fields (`sim1_data[]` / `sim2_data[]` / `dual_data[]`):

Field	Type	Description
`timestamp`	number	Unix timestamp (seconds). The `day` dimension is aligned by hour, the `month` dimension is aligned by day
`rx_bytes`	number	Downstream traffic (bytes)
`tx_bytes`	number	Upstream traffic (bytes)

1.3 Network

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the request header `x-session-id: <td>`.

Top-level response fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version number
`status`	number	Request execution status: `0` success

1.3.1 Get Interface List

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_netflow",
  "function": "get",
  "values": [{ "type": "if_list" }]
```

```
}
```

Response


```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "if_list",
          "value": {
            "interfaces": [
              "WAN",
              "WLAN(Client)",
              "Bridge0",
              "vlan1",
              "vlan4",
              "vlan2",
              "vlan3",
              "dmvpn"
            ]
          }
        }
      ]
    }
  ]
}
```


Field Description

Field	Type	Description
`interfaces`	string[]	List of available network interface names

1.3.2 Get Interface Status

Request

```
{
  "id": 2,
  "execute": 1,
  "core": "yruo_netflow",
  "function": "get",
  "values": [{ "type": "if_status", "interface": "WAN" }]
}
```

Parameter	Type	Required	Description
`interface`	string	Yes	Interface name, obtained from the interface list

Response


```
{
  "id": 2,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "if_status",
          "interface": "WAN",
          "value": {
            "interface": "WAN",
            "dial_type": "static",
            "ipv4": {
              "ip": "192.168.50.3",
              "netmask": "255.255.255.0",
              "gateway": "192.168.50.1",
              "dns": "192.168.1.1",
              "sec_dns": "8.8.8.8"
            },
            "ipv6": {
              "ip": "2001:db8:85a3::8a2e:370:7334",
              "prefix_len": 64,
              "gateway": "-",
              "dns": "",
              "sec_dns": ""
            },
            "rx_bytes": 2149618368,
            "tx_bytes": 102053347
          }
        }
      ]
    }
  ]
}
```


Field Description

`get[]` entry fields:

Field	Type	Description
`type`	string	Fixed as `if_status`
`interface`	string	Name of the queried interface

`value` object:

Field	Type	Description
`interface`	string	Interface name
`dial_type`	string	Dial type: `dhcp` / `dhcpv6` / `static` / `pppoe` / `dual_stack` / `none`
`ipv4.ip`	string	IPv4 address
`ipv4.netmask`	string	Netmask
`ipv4.gateway`	string	IPv4 gateway
`ipv4.dns`	string	IPv4 primary DNS
`ipv4.sec_dns`	string	IPv4 secondary DNS
`ipv6.ip`	string	IPv6 address
`ipv6.prefix_len`	number	IPv6 prefix length
`ipv6.gateway`	string	IPv6 gateway, `"-"` when unavailable
`ipv6.dns`	string	IPv6 primary DNS, empty string when unavailable
`ipv6.sec_dns`	string	IPv6 secondary DNS, empty string when unavailable
`rx_bytes`	number	Cumulative downstream traffic (bytes)
`tx_bytes`	number	Cumulative upstream traffic (bytes)

1.3.3 Get Realtime Speed

Poll every 3 seconds; returns the receive/send speed data point at the current moment.

Request

```
{
  "id": 4,
  "execute": 1,
  "core": "yruo_netflow",
  "function": "get",
  "values": [{ "type": "netflow", "interface": "WAN", "time_range": "realtime" }]
}
```

Parameter	Type	Required	Description
`interface`	string	Yes	Interface name

Response


```
{
  "id": 4,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "netflow",
          "interface": "WAN",
          "time_range": "realtime",
          "value": {
            "timezone": "+0100",
            "timezone_offset": "0",
            "data_points": [
              {
                "timestamp": 1775726999,
                "rx_bps": 31432,
                "tx_bps": 0
              }
            ]
          }
        }
      ]
    }
  ]
}
```


Field Description

`get[]` entry fields:

Field	Type	Description
`type`	string	Fixed as `netflow`
`interface`	string	Name of the queried interface
`time_range`	string	Consistent with the request, here `realtime`

`value` object:

Field	Type	Description
`timezone`	string	Device timezone offset, e.g. `"+0100"`
`timezone_offset`	string	Additional timezone offset (daylight saving time, etc.), usually `"0"`

Field	Type	Description
`data_points[].timestamp`	number	Unix timestamp (seconds)
`data_points[].rx_bps`	number	Downstream speed (bps)
`data_points[].tx_bps`	number	Upstream speed (bps)

1.3.4 Get Traffic History

Query traffic data aggregated by hour/day/month over a historical time period.

Request

```
{
  "id": 3,
  "execute": 1,
  "core": "yruo_netflow",
  "function": "get",
  "values": [{ "type": "netflow", "interface": "WAN", "time_range": "day" }]
}
```

Parameter	Type	Required	Description
`interface`	string	Yes	Interface name
`time_range`	string	Yes	Time range: `day` (today, aggregated by hour) / `month` (this month, aggregated by day) / `year` (this year, aggregated by month)

Response

The response structure is the same as 1.3.3. The `time_range` in the `get[]` entry is the value passed in the request (`day` / `month` / `year`); `value` does not contain `timezone` / `timezone_offset`, and `data_points` is the aggregated data for the corresponding time granularity.

```
{
  "id": 3,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "netflow",
          "interface": "WAN",
          "time_range": "day",
          "value": {
```

```
            "data_points": [
              { "timestamp": 1775692800, "rx_bytes": 10485760, "tx_bytes":
5242880 },
              { "timestamp": 1775696400, "rx_bytes": 8388608, "tx_bytes": 4194304
}
            ]
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`data_points[].timestamp`	number	Unix timestamp (seconds), start time of the period
`data_points[].rx_bytes`	number	Downstream traffic within the period (bytes)
`data_points[].tx_bytes`	number	Upstream traffic within the period (bytes)

1.4 Routing

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the request header `x-session-id: <td>`.

1.4.1 Get Routing and ARP Information

Request

```
{
  "id": 5,
  "execute": 1,
  "core": "yruo_status",
  "function": "get",
  "values": [{ "base": "yruo_status_route" }]
}
```

Response

```
{
  "id": 5,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
```

```
{
  "get": [
    {
      "type": "yruo_status_route",
      "index": 1,
      "value": {
        "dest": "0.0.0.0",
        "mask": "0.0.0.0",
        "gate": "192.168.50.1",
        "ifn": "WAN/LAN5",
        "distance": 1
      }
    },
    {
      "type": "yruo_status_route",
      "index": 2,
      "value": {
        "dest": "192.168.1.0",
        "mask": "255.255.255.0",
        "ifn": "Bridge0"
      }
    },
    {
      "type": "yruo_status_route",
      "index": 3,
      "value": {
        "dest": "::",
        "mask": "0",
        "ifn": "SIM1-APN1"
      }
    },
    {
      "type": "yruo_status_route",
      "index": 4,
      "value": {
        "dest": "2409:8934:20dc:e5b::",
        "mask": "64",
        "ifn": "SIM1-APN1"
      }
    },
    {
      "type": "yruo_status_arp",
      "index": 1,
      "value": {
        "ip": "192.168.50.1",
        "mac": "b8:e3:b1:90:fd:06",
        "ifn": "WAN/LAN5"
      }
    },
    {
      "type": "yruo_status_arp",
      "index": 2,
      "value": {
        "ip": "192.168.50.2",
        "mac": "14:98:77:7f:4c:87",
        "ifn": "WAN/LAN5"
```

```
          }
        }
      ]
    }
  ]
}
```

Field Description

The `result[0].get` array in the response contains two types of data, distinguished by the `type` field:

- `yruo_status_route`: Routing table entries (IPv4 and IPv6 mixed)
- `yruo_status_arp`: ARP cache entries

`yruo_status_route` value object

Field	Type	Description
`dest`	string	Destination address, IPv4 dotted decimal or IPv6 address
`mask`	string	Netmask. For IPv4 it is dotted decimal (e.g. `"255.255.255.0"`); for IPv6 it is a prefix length numeric string (e.g. `"64"`)
`gate`	string	Gateway address; directly connected routes do not include this field
`ifn`	string	Outgoing interface name, e.g. `WAN/LAN5`, `SIM1-APN1`, `Bridge0`, `dmvpn`, `Loopback`
`distance`	number	Administrative distance; directly connected routes do not include this field

`yruo_status_arp` value object

Field	Type	Description
`ip`	string	Neighbor IP address
`mac`	string	Corresponding MAC address
`ifn`	string	Associated interface name

1.5 VPN

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the request header `x-session-id: <td>`.

1.5.1 Get VPN Status

Request

```
{
  "id": 10,
  "execute": 1,
  "core": "yruo_vpn_status",
  "function": "get",
  "values": [{ "base": "yruo_vpn_status" }]
}
```

Response


```
{
  "id": 10,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "clients",
          "index": 1,
          "value": {
            "name": "dmvpn",
            "status": 2,
            "local_ip": "10.0.0.1",
            "remote_ip": "10.0.0.2"
          }
        },
        {
          "type": "server",
          "index": 2,
          "value": {
            "id": 1,
            "name": "OpenVPN Server",
            "status": 40
          }
        },
        {
          "type": "server",
          "index": 3,
          "value": {
            "id": 1,
            "name": "Ipsec Server",
            "status": 40
          }
        },
        {
          "type": "connected_list",
```

```
          "index": 4,
          "value": {
            "type": "OpenVPN",
            "remote_ip": "192.168.1.100",
            "connect_time": "01:23:45"
          }
        }
      ]
    }
  ]
}
```


Field Description

The `result[0].get` array in the response contains three types of data, distinguished by the `type` field:

- `clients`: VPN client entries, one for each configured VPN client
- `server`: VPN server entries, one for each configured VPN server
- `connected_list`: Remote clients connected to the server, one for each online connection; this type of entry is not returned when there are no connections

`clients` value object

Field	Type	Description
`name`	string	VPN client name, e.g. `dmvpn`, `openvpn`
`status`	number	Connection status, enum values see table below
`local_ip`	string	Local VPN IP address, not returned when not connected
`remote_ip`	string	Remote VPN IP address, not returned when not connected

`clients.status` enum values:

Value	Color	Description
`2`	Gray	Not connected
`3`	Green	Connected
`400`–`405`	Gray	Error / various not-connected states

`server` value object

Field	Type	Description
`id`	number	Server instance ID
`name`	string	Server name, e.g. `OpenVPN Server`, `Ipsec Server`
`status`	number	Running status, enum values see table below

`server.status` enum values:

Value	Color	Description
`40`	Gray	Disabled / not running
`41`	Green	Running

`connected_list` value object

Field	Type	Description
`type`	string	VPN type, e.g. `OpenVPN`, `IPsec`
`remote_ip`	string	Connected remote client IP
`connect_time`	string	Connected duration

1.6 Host List

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the request header `x-session-id: <td>`.

Top-level response fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version number
`status`	number	Request execution status: `0` success

1.6.1 Get Host List

Returns the current DHCP lease list and ARP host list; `get` is an empty array when there are no entries.

Request

```
{
  "id": 6,
  "execute": 1,
  "core": "yruo_status",
  "function": "get",
  "values": [{ "base": "yruo_status_dhcp" }]
}
```

Response

```
{
  "id": 6,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_status_dhcplease",
          "index": 0,
          "value": {
            "ip": "192.168.1.100",
            "mac": "AA:BB:CC:DD:EE:FF",
            "lease": 86400
          }
        },
        {
          "type": "yruo_status_dhcplease",
          "index": 1,
          "value": {
            "ip": "192.168.1.101",
            "mac": "11:22:33:44:55:66",
            "lease": "-"
          }
        },
        // ... more lease entries
        {
          "type": "yruo_status_dhcphost",
          "index": 0,
          "value": {
            "ip": "192.168.1.200",
            "mac": "AA:11:BB:22:CC:33"
          }
        },
        {
          "type": "yruo_status_dhcphost",
          "index": 1,
          "value": {
            "ip": "192.168.1.201",
```

```
            "mac": "DD:44:EE:55:FF:66"
          }
        }
        // ... more host entries
      ]
    }
  ]
}
```

Field Description

The `result[0].get` array in the response contains two types of data, distinguished by the `type` field:

- `yruo_status_dhcplease`: DHCP lease entries, one for each active lease
- `yruo_status_dhcphost`: ARP host entries, one for each known host

`yruo_status_dhcplease` value object (DHCP lease)

Field	Type	Description
`ip`	string	Assigned IPv4 address
`mac`	string	Client MAC address
`lease`	number | string	Remaining lease time (seconds); if the string `"-"`, it means never expires

`lease` display conversion rules:

Value	Frontend display
`"-"`	Never expires (unlimited)
Number (seconds)	Converted to `D days, HH hours MM minutes SS seconds` format, omitting the highest unit that is zero

`yruo_status_dhcphost` value object (ARP host)

Field	Type	Description
`ip`	string	Host IPv4 address
`mac`	string	Host MAC address

1.7 GPS

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the request header `x-session-id: <td>`.

Top-level response fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

1.7.1 Query GPS Status

This interface is a read-only status query with no corresponding save interface. The frontend page polls this interface periodically to refresh the display.

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_industrial_gps_status",
  "values": [
    { "base": "yruo_industrial_gps_status" }
  ]
}
```

Response

Locating (`status`=`"51"`):

```
{
  "id": 12,
  "model": "UR41",
  "pn": "L08EU0011111DE0000000000",
  "oem": "0000",
  "rtver": "41.0.0.7-a2",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_industrial_gps_status",
          "index": 1,
          "value": {
            "status": "51"
          }
        }
```

```
      ]
    }
  ]
}
```

Located (`status`=`"53"`, complete fields, inferred from code):

```
{
  "result": [
    {
      "get": [
        {
          "type": "yruo_industrial_gps_status",
          "index": 1,
          "value": {
            "status": "53",
            "time": "2024-01-01 12:00:00",
            "sats": "8",
            "sativ": "12",
            "latitude": "22.543096",
            "longitude": "114.057865",
            "height": "10.0",
            "speed": "0.0"
          }
        }
      ]
    }
  ]
}
```

> The `time`, `sats`, `sativ`, `latitude`, `longitude`, `height`, and `speed` fields are only returned when GPS is located (`status`=`"53"`); while locating, the response `value` only contains the `status` field (inferred from actual packet captures, not fully verified across all states).

Field Description

`yruo_industrial_gps_status` type (type=`yruo_industrial_gps_status`, index=`1`)

Field	Type	Description
`status`	string	GPS location status, see enum table
`time`	string	GPS time, returned when located
`sats`	string	Number of available satellites, returned when located
`sativ`	string	Number of visible satellites, returned when located
`latitude`	string	Latitude, returned when located
`longitude`	string	Longitude, returned when located
`height`	string	Altitude (meters), returned when located

Field	Type	Description
`speed`	string	Speed (km/h), returned when located

`status` enum values (known enum values, may be incomplete)

Value	UI color	Meaning
`"50"`	Gray	No signal / initializing
`"51"`	Gray	Locating (searching for satellites)
`"53"`	Green	Located

1.8 WLAN

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the request header `x-session-id: <td>`.

1.8.1 Get WLAN Status

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_wifi_status",
  "function": "get",
  "values": [{ "base": "yruo_wifi_status" }]
}
```

Response

```
{
  "id": 8,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_wifi_status",
          "index": "0",
          "value": {
            "interface": "WLAN",
            "status": 30,
            "mode": 0,
            "ssid": "Router_FEB735",
```

```
            "ip": "192.168.1.1",
            "ipv6": "",
            "mask": "255.255.255.0"
          }
        },
        {
          "type": "yruo_wifi_status",
          "index": "1",
          "value": {
            "interface": "WLAN1",
            "status": 33,
            "mode": 1,
            "ssid": "UpstreamAP",
            "ip": "192.168.0.100",
            "ipv6": "",
            "mask": "255.255.255.0"
          }
        },
        {
          "type": "yruo_wifi_ass",
          "index": "0",
          "value": {
            "interface": "WLAN",
            "mac": "AA:BB:CC:DD:EE:FF",
            "ip": "192.168.1.100",
            "ipv6": "",
            "connected": "3600 0"
          }
        },
        {
          "type": "yruo_wifi_ass",
          "index": "1",
          "value": {
            "interface": "WLAN",
            "mac": "11:22:33:44:55:66",
            "ip": "192.168.1.101",
            "ipv6": "fe80::2",
            "connected": "90061 0"
          }
        }
      ]
    }
  ]
}
```


Field Description

Top-level response fields

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model

Field	Type	Description
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version number
`status`	number	Request execution status: `0` success

The `result[0].get` array in the response contains two types of data, distinguished by the `type` field:

- `yruo_wifi_status`: WLAN interface status, one for each wireless interface
- `yruo_wifi_ass`: Associated wireless clients, one for each client

`yruo_wifi_status` value object (WLAN interface status)

Field	Type	Description
`interface`	string	Interface name, e.g. `wlan0`, `wlan1`
`status`	number	Interface status, enum values see table below
`mode`	number	Operating mode: `0` AP (hotspot), `1` STA (client)
`ssid`	string	SSID name
`ip`	string	IPv4 address
`mask`	string	Netmask, used to compute the CIDR prefix length
`ipv6`	string	IPv6 address, empty string if none

`status` enum values:

Value	Description
`30`	Connected
`31`	Disabled
`32`	Connecting
`33`	Associated (STA mode connected to upstream AP)
`34`	Error / disabled

`yruo_wifi_ass` value object (associated wireless client)

Field	Type	Description
`interface`	string	Associated interface name
`mac`	string	Client MAC address

Field	Type	Description
`ip`	string	Client IPv4 address
`ipv6`	string	Client IPv6 address, empty string if none
`connected`	string	Connected duration, format `"<seconds> <extra value>"`; take the number before the space (seconds) for time-formatted display

2. Network

2.1 Interface

2.1.1 Link Backup

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

Get Link Backup Configuration

Request

```
{
  "id": 42,
  "function": "get",
  "core": "yruo_if_backup",
  "values": [{ "base": "yruo_if_backup" }]
}
```

Response

```
{
  "id": 42,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
```

```
"rtver": "35.3.0.12-a1",
"status": 0,
"result": [
  {
    "get": [
      {
        "type": "yruo_if_backup",
        "index": "qwerty",
        "value": {
          "links": [
            {
              "priority": 1,
              "enable": 1,
              "current": 1,
              "type": 0,
              "enable_icmp": 1,
              "icmp_period": 300,
              "icmp_interval": 5,
              "icmp_timeout": 3,
              "icmp_times": 3,
              "interface": "WAN",
              "ip": "192.168.50.3",
              "dest_main": "8.8.8.8",
              "dest_second": "223.5.5.5",
              "dest6_main": "2001:4860:4860::8888",
              "dest6_second": "2400:3200::1"
            },
            {
              "priority": 2,
              "enable": 1,
              "current": 0,
              "type": 1,
              "enable_icmp": 1,
              "icmp_period": 300,
              "icmp_interval": 5,
              "icmp_timeout": 3,
              "icmp_times": 3,
              "interface": "SIM2-APN1",
              "ip": "",
              "dest_main": "8.8.8.8",
              "dest_second": "223.5.5.5",
              "dest6_main": "2001:4860:4860::8888",
              "dest6_second": "2400:3200::1"
            }
            // ...more link entries
          ],
          "resume_enable": 1,
          "resume_interval": 300,
          "sim_revert_enable": 0,
          "sim_revert_interval": 1,
          "sim_delay_change_interval": 0,
          "error_reboot": 0
        }
      }
    ]
  }
```

```
  ]
}
```

yruo_if_backup Field Descriptions

Top-level fields

Field	Type	Description
`links`	array	Link priority list, up to 5 entries, sorted ascending by `priority`
`resume_enable`	integer	Link failover recovery, `0`=disabled, `1`=enabled
`resume_interval`	integer	Recovery detection interval (seconds), 0–604800; takes effect when `resume_enable=1`
`sim_delay_change_interval`	integer	Dual-SIM switch delay (seconds), 0–3600; dual-SIM devices only, takes effect when `resume_enable=1`
`sim_revert_enable`	integer	Dual-SIM recovery, `0`=disabled, `1`=enabled; dual-SIM devices only, takes effect when `resume_enable=1`
`sim_revert_interval`	integer	Dual-SIM recovery interval (minutes), 1–1440; dual-SIM devices only, takes effect when `sim_revert_enable=1`
`error_reboot`	integer	Auto reboot when all links fail, `0`=disabled, `1`=enabled

links entry fields

Field	Type	Description
`priority`	integer	Priority; the smaller the number, the higher the priority
`enable`	integer	Whether this link rule is enabled, `0`=disabled, `1`=enabled (at least one must be enabled)
`current`	integer	Whether currently the active link (read-only), `0`=no, `1`=yes
`type`	integer	Link type (read-only, known enum values, may be incomplete), `0`=WAN, `1`=Cellular
`interface`	string	Interface name (read-only), e.g. `"WAN"`, `"SIM1-APN1"`
`ip`	string	Current IP address (read-only), empty string when not obtained
`enable_icmp`	integer	Whether ICMP probing is enabled, `0`=disabled, `1`=enabled
`dest_main`	string	Primary probe target IPv4 address

Field	Type	Description
`dest_second`	string	Secondary probe target IPv4 address
`dest6_main`	string	Primary probe target IPv6 address (only effective for interfaces that support IPv6)
`dest6_second`	string	Secondary probe target IPv6 address (only effective for interfaces that support IPv6)
`icmp_period`	integer	ICMP probe period (seconds), 10–1800
`icmp_interval`	integer	ICMP probe interval (seconds), 3–600
`icmp_timeout`	integer	ICMP timeout (seconds), 1–10
`icmp_times`	integer	ICMP probe failure threshold, 1–10

Save Link Backup Configuration

Request


```
{
  "id": 43,
  "function": "set",
  "core": "yruo_if_backup",
  "values": [
    {
      "base": "yruo_if_backup",
      "type": "yruo_if_backup",
      "index": "qwerty",
      "value": {
        "links": [
          {
            "priority": 1,
            "enable": 1,
            "current": 1,
            "type": 0,
            "enable_icmp": 1,
            "icmp_period": 300,
            "icmp_interval": 5,
            "icmp_timeout": 3,
            "icmp_times": 3,
            "interface": "WAN",
            "ip": "192.168.50.3",
            "dest_main": "8.8.8.8",
            "dest_second": "223.5.5.5",
            "dest6_main": "2001:4860:4860::8888",
            "dest6_second": "2400:3200::1"
          }
        ],
        "resume_enable": 1,
        "resume_interval": 300,
        "sim_revert_enable": 0,
        "sim_revert_interval": 1,
```

```
        "sim_delay_change_interval": 0,
        "error_reboot": 0
      }
    }
  ]
}
```

> The `links` array must be submitted as the full list (including order). The front end reorders via drag-and-drop and then submits in the new order. At least one link rule with `enable=1` is required, otherwise the save is rejected.

Response

```
{
  "id": 43,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

2.1.2 Cellular

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request sequence number
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM code
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.1.2.1 Query Cellular Configuration

Request

```
{
  "core": "yruo_cell",
  "function": "get",
  "values": [{ "base": "yruo_cell" }]
}
```

Response


```
{
  "id": 43,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_cell",
          "index": 0,
          "value": {
            "phone_groups": [3],
            "need_reboot": 1,
            "networks": 7,
            "revision": "EC25EUXGAR08A17M1G_A0.200.A0.200",
            "support_lte_band": [1, 3, 7, 8, 20, 28, 38, 40, 41],
            "sim1_network": [
              {
                "interface_name": "SIM1-APN1",
                "status": 1,
                "enable": 1,
                "type": 0,
                "ip": "10.185.18.200",
                "apn": "cmnet",
                "user": "",
                "password": "",
                "auth": 0,
                "nat": 1,
                "protocol": 0,
                "mtu": 1500,
                "forced_mtu": 0,
                "pri_dns": "",
                "sec_dns": "",
                "pri_v6dns": "",
                "sec_v6dns": "",
                "ppp_prefer0": 0,
                "auto_apn": 1
              },
              {
                "interface_name": "SIM1-APN2",
```

```
      "status": -1,
      "enable": 0,
      "type": 0,
      "ip": "-",
      "apn": "",
      "user": "",
      "password": "",
      "auth": 0,
      "nat": 1,
      "protocol": 0,
      "mtu": 1500,
      "forced_mtu": 0,
      "pri_dns": "",
      "sec_dns": "",
      "pri_v6dns": "",
      "sec_v6dns": ""
    }
    // APN3 structure same as APN2, omitted
  ],
  "sim2_network": [
    // 3 entries, structure same as sim1_network, omitted
  ],
  "network1": 0,
  "network2": 0,
  "pin1": "",
  "pin2": "",
  "plmn1": "",
  "plmn2": "",
  "number1": "",
  "number2": "",
  "ims_enable1": 1,
  "ims_enable2": 1,
  "sms_center1": "",
  "sms_center2": "",
  "roam1": 1,
  "roam2": 1,
  "mask1": "",
  "mask2": "",
  "traffic_limit1": 0,
  "traffic_limit2": 0,
  "threshold1": 1,
  "threshold2": 1,
  "switch_card_traffic1": 1,
  "switch_card_traffic2": 1,
  "billing_day1": 1,
  "billing_day2": 1,
  "sms_limit1": 0,
  "sms_limit2": 0,
  "max_sms1": 1,
  "max_sms2": 1,
  "switch_card_sms1": 1,
  "switch_card_sms2": 1,
  "switch_card_weak_signal1": 0,
  "switch_card_weak_signal2": 0,
  "signal_strength1": "-105",
  "signal_strength2": "-105",
```

```
            "check_interval1": 30,
            "check_interval2": 30,
            "check_count1": 3,
            "check_count2": 3,
            "lock_lte_band1": [1, 3, 7, 8, 20, 28, 38, 40, 41],
            "lock_lte_band2": [1, 3, 7, 8, 20, 28, 38, 40, 41],
            "conn_mode": 0,
            "redial_interval": 5,
            "max_idle": 60,
            "trigger_call": 0,
            "trigger_sms": 0,
            "trigger_io": 0,
            "dialtimeout": 5,
            "volte": 1,
            "hotline": 1,
            "onlyhotline": 0,
            "counry": "us",
            "feturedigit1": "",
            "dialnumber1": "",
            "feturedigit2": "",
            "dialnumber2": "",
            "feturedigit3": "",
            "dialnumber3": "",
            "txgain": 20577,
            "txdgain": 14567,
            "rxgain": 20577,
            "clvl": 3
          }
        }
      ]
    }
  ]
}
```


> **Note**: The response also contains legacy top-level fields (see the "Legacy Fields" table below). These fields are not used in the current UI version; the actual APN configuration is governed by the `sim1_network` / `sim2_network` arrays.

Field Descriptions

value top-level fields

Field	Type	Read-only	Description
`phone_groups`	number[]	✅	List of available phone group IDs
`need_reboot`	number	✅	Whether the first multi-APN enable requires a reboot: `0`=no, `1`=yes
`networks`	number	✅	Bitmask of supported network types: bit0=2G, bit1=3G, bit2=4G
`revision`	string	✅	Module firmware version number

Field	Type	Read-only	Description
`support_lte_band`	number[]	✅	List of LTE band numbers supported by the module

sim{n}_network array items (up to 3 APNs per SIM card, `{n}` is `1` or `2`)

Field	Type	Read-only	Description
`interface_name`	string	✅	Interface name, e.g. `"SIM1-APN1"`
`status`	number	✅	Connection status, see status enum
`type`	number	✅	Current network type (synced from `network{n}`), see network type enum
`ip`	string	✅	Current IP address, `"-"` when not connected
`enable`	number		Whether enabled: `0` / `1`
`apn`	string		Access point name, max 63 characters
`user`	string		Username, max 63 characters
`password`	string		Password, max 63 characters; GET always returns empty string `""`, SET sends plaintext
`auth`	number		Authentication method: `0`=None, `1`=PAP, `2`=CHAP
`nat`	number		Enable NAT: `0` / `1`
`protocol`	number		Protocol type: `0`=IPv4, `1`=IPv6, `2`=IPv4/IPv6
`mtu`	number		MTU, IPv4 range 68–1500, IPv6/dual-stack range 1280–1500
`forced_mtu`	number		Enable custom MTU: `0` / `1`
`pri_dns`	string		Custom primary IPv4 DNS
`sec_dns`	string		Custom secondary IPv4 DNS
`pri_v6dns`	string		Custom primary IPv6 DNS
`sec_v6dns`	string		Custom secondary IPv6 DNS
`ppp_prefer0`	number		PPP preference: `0` / `1`; **only present on APN1 (index 0)**
`auto_apn`	number		Auto APN: `0` / `1`; **only present on APN1 (index 0)**

Per-SIM card configuration fields (`{n}` is `1` or `2`)

Field	Type	Description
`network{n}`	number	Network type, see network type enum
`pin{n}`	string	SIM card PIN, max 31 characters
`plmn{n}`	string	Operator-locked PLMN code, allows 3, 5, or 6 digits, empty string means not locked
`number{n}`	string	Dial center number, max 31 characters
`ims_enable{n}`	number	Enable IMS/VoLTE: `0` / `1`
`sms_center{n}`	string	SMS center number, max 20 characters, format: digits only or `+` followed by digits
`roam{n}`	number	Allow roaming: `0` / `1`
`mask{n}`	string	IPv4 netmask (this field exists only on some firmware versions)
`traffic_limit{n}`	number	Enable traffic limit: `0` / `1`
`threshold{n}`	number	Traffic cap (MB), takes effect when `traffic_limit{n}=1`
`switch_card_traffic{n}`	number	Switch SIM after traffic cap reached: `0` / `1`; **dual-SIM devices only**
`billing_day{n}`	number	Monthly billing settlement day, range 1–28
`sms_limit{n}`	number	Enable SMS limit: `0` / `1`
`max_sms{n}`	number	SMS cap, range 1–10000
`switch_card_sms{n}`	number	Switch SIM after SMS cap reached: `0` / `1`; **dual-SIM devices only**
`switch_card_weak_signal{n}`	number	Switch SIM on weak signal: `0` / `1`; **dual-SIM devices only**
`signal_strength{n}`	string	Weak-signal threshold (dBm), integer string, range `-120` – `-90`; takes effect when `switch_card_weak_signal{n}=1`, **dual-SIM devices only**
`check_interval{n}`	number	Weak-signal check interval (seconds), range 20–3600; **dual-SIM devices only**
`check_count{n}`	number	Weak-signal check count, range 1–10; **dual-SIM devices only**
`lock_lte_band{n}`	number[]	List of enabled locked LTE bands, elements taken from `support_lte_band`

Connection settings fields

Field	Type	Description
`conn_mode`	number	Connection mode: `0`=always-on, `1`=on-demand
`redial_interval`	number	Redial interval (seconds), range 0–3600
`max_idle`	number	Maximum idle time (seconds), range 10–3600; takes effect when `conn_mode=1`
`trigger_call`	number	Dial on incoming call: `0` / `1`; takes effect when `conn_mode=1`
`trigger_sms`	number	Dial on SMS: `0` / `1`; takes effect when `conn_mode=1`
`trigger_io`	number	Dial on IO trigger: `0` / `1`; takes effect when `conn_mode=1`, not supported on UR32L/UR32S

VoIP settings fields (only on UR35 models with the VoIP module)

Field	Type	Read-only	Description
`hotline`	number	√	Whether the hotline feature is available
`volte`	number		Enable VoLTE: `0` / `1`
`dialtimeout`	number		Dial timeout (seconds), range 5–60
`counry`	string		VoIP country/region, see country enum
`feturedigit1` / `feturedigit2` / `feturedigit3`	string		Feature dial prefix, max 7 characters
`dialnumber1` / `dialnumber2` / `dialnumber3`	string		Corresponding hotline number, max 32 characters; required when `feturedigit` is non-empty
`onlyhotline`	number		Hotline-only mode: `0` / `1`; when enabled, at least one `feturedigit` group must be configured
`txgain`	number		Transmit gain, range 0–65535
`txdgain`	number		Transmit digital gain, range 0–65535
`rxgain`	number		Receive gain, range 0–65535
`clvl`	number		Call volume level, range 0–5

Connection status enum (`sim{n}_network[].status`)

Value	Description	Display color
`-1`	N/A (not activated)	shown as `"-"`
`0`	Connection failed	Red
`1`	Connection successful	Green

Network type enum (`sim{n}_network[].type`, `network{n}`)

Value	Description
`0`	4G (auto)
`2`	4G Only
`4`	3G
`6`	2G

> The actual selectable values depend on the `networks` bitmask; the device only shows options supported by the hardware.

`counry` enum (partial, known enum values)

Value	Description
`us`	United States / North America
`cn`	China
`uk`	United Kingdom
`de`	Germany
`fr`	France
`jp`	Japan
`kr`	Korea
`au`	Australia
`in`	India
`br`	Brazil
`ru`	Russian Federation
`custom`	User custom for Tone Zone

> There are 43 options in total (including major countries/regions in Europe, Asia, the Americas, etc.); known enum values, may be incomplete.

Legacy top-level fields

The response also contains the following top-level fields, retained for compatibility with legacy firmware; the current UI version now manages APN configuration via the `sim{n}_network` arrays.

Field	Description
`protocol{n}`	Legacy protocol field
`apn{n}`, `user{n}`, `password{n}`	Legacy APN authentication fields
`auth{n}`, `ppp_prefer{n}`	Legacy authentication/PPP fields
`nat{n}`	Legacy NAT field
`forced_mtu{n}`, `mtu{n}`	Legacy MTU fields
`pri_dns{n}`, `sec_dns{n}`, `pri_v6dns{n}`, `sec_v6dns{n}`	Legacy DNS fields
`oslectaps`	Internally reserved field

2.1.2.2 Save Cellular Configuration

Request

```
{
  "core": "yruo_cell",
  "function": "set",
  "values": [
    {
      "base": "yruo_cell",
      "type": "yruo_cell",
      "index": 0,
      "value": {
        "sim1_network": [
          {
            "interface_name": "SIM1-APN1",
            "status": 1,
            "enable": 1,
            "type": 0,
            "ip": "10.185.18.200",
            "apn": "cmnet",
            "user": "",
            "password": "mypassword",
            "auth": 0,
            "nat": 1,
            "protocol": 0,
            "mtu": 1500,
            "forced_mtu": 0,
            "pri_dns": "",
            "sec_dns": "",
            "pri_v6dns": "",
```

```
            "sec_v6dns": "",
            "ppp_prefer0": 0,
            "auto_apn": 1
          }
          // APN2, APN3 as above (without ppp_prefer0/auto_apn)
        ],
        "sim2_network": [
          // structure as above; may be omitted on single-SIM devices
        ],
        "network1": 0,
        "pin1": "",
        "plmn1": "",
        "number1": "",
        "ims_enable1": 1,
        "sms_center1": "",
        "roam1": 1,
        "traffic_limit1": 0,
        "threshold1": 1,
        "billing_day1": 1,
        "sms_limit1": 0,
        "max_sms1": 1,
        "lock_lte_band1": [1, 3, 7, 8, 20, 28, 38, 40, 41],
        "conn_mode": 0,
        "redial_interval": 5
      }
    }
  ]
}
```

> - `sim1_network` / `sim2_network` must be submitted as the **full array** (including unmodified entries), in the current order.
> - `lock_lte_band{n}` is an integer array, same format as GET, without the `"B"` prefix.
> - Password fields (`password`, `pin{n}`) send plaintext on SET; if the password is unchanged, an empty string or the original value may be sent.
> - **is4X devices only**: On save, an additional `yruo_if_backup` SET request is triggered to write the ICMP ping probe configuration (`enable_icmp`, `dest_main`, `dest_second`, `dest6_main`, `dest6_second`, `icmp_period`, `icmp_interval`, `icmp_timeout`, `icmp_times`) and `error_reboot` into the corresponding link of Link Backup; see 2.1.1 Link Backup.

2.1.3 Port

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request sequence number
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM code
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.1.3.1 Query Port Configuration

Request

```
{
  "core": "yruo_port",
  "function": "get",
  "values": [{ "base": "yruo_port" }]
}
```

Response


```
{
  "id": 44,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_port",
          "index": "Lk3TCI784dP4937RA2o9l82J8lGpR67OtiYIZalYHU3cK",
          "value": {
            "port": "WAN/LAN5",
            "status": 1,
            "property": 2,
            "speed": 0,
            "duplex": 0,
            "conferred": 0,
            "connect_state": 1,
            "cur_speed": "100Mbps",
            "cur_duplex": "full"
          }
        },
        {
```

```
          "type": "yruo_port",
          "index": "8Ja419ovhC0pykg09pNoKlp4lK1O0dyhSH530ZK369wB",
          "value": {
            "port": "LAN1",
            "status": 1,
            "property": 1,
            "speed": 0,
            "duplex": 0,
            "conferred": 0,
            "connect_state": 0,
            "cur_speed": "-",
            "cur_duplex": "-"
          }
        },
        // LAN2, LAN3, LAN4 have the same structure as LAN1, omitted
        {
          "type": "yruo_8021x",
          "index": "1aAbB2cCdDeE3",
          "value": {
            "ieee802_1x": 0,
            "auth_addr": "",
            "auth_port": 1812,
            "auth_secret": "",
            "nas_id": ""
          }
        }
      ]
    }
  ]
}
```

> - Each physical port corresponds to one `yruo_port` record; each port's `index` is an opaque string assigned by the device and must be passed back unchanged on save.
> - PoE devices also include `type: "yruo_poe"` records; see the PoE field descriptions.
> - `yruo_8021x` is returned only on devices that support 802.1X authentication.

Field Descriptions

yruo_port (port configuration, one per port)

Field	Type	Read-only	Description
`port`	string	✅	Port name, e.g. `"WAN/LAN5"`, `"LAN1"`
`connect_state`	number	✅	Physical link state: `0`=not connected (gray), `1`=connected (green)
`cur_speed`	string	✅	Current negotiated speed, e.g. `"100Mbps"`, `"-"` when not connected
`cur_duplex`	string	✅	Current negotiated duplex mode, e.g. `"full"`, `"-"` when not connected

Field	Type	Read-only	Description
`conferred`	number	✅	Negotiation-complete flag (internal use only, not displayed)
`status`	number		Port administrative state: `0`=down, `1`=up
`property`	number		Port property: `1`=lan, `2`=wan; **only modifiable on ports whose name contains both WAN and LAN**; this field is not shown on is4X devices
`speed`	number		Speed setting, see speed enum
`duplex`	number		Duplex setting: `0`=auto, `1`=half, `2`=full

yruo_poe (PoE configuration, PoE devices only, index: `"qwerty"`)

`value.poe[]` is the PoE port list, sorted ascending by `priority`.

Field	Type	Read-only	Description
`port`	string	✅	Port name
`power_status`	number	✅	Power state: `0`=not powered (gray), `1`=powering (green)
`voltage`	number	✅	Current voltage (V)
`current`	number	✅	Current current (A)
`power`	string	✅	Current power (W)
`priority`	number		Priority; reorder in the table via move up/down
`power_enable`	number		PoE power switch: `0`=off, `1`=on
`description`	string		Description, max 63 characters, must not contain spaces, `;`, `(`
`icmp_enable`	number		Enable Ping probing: `0` / `1`
`icmp_dest`	string		Ping target IPv4 address, takes effect when `icmp_enable=1`
`icmp_interval`	number		Ping check period (minutes), range 5–1440
`icmp_retry_interval`	number		Ping retry interval (seconds), range 3–600
`icmp_timeout`	number		Ping timeout (seconds), range 1–10
`icmp_times`	number		Ping timeout count, range 1–10

Field	Type	Read-only	Description
`power_cut_interval`	number		Power-off reboot wait time (seconds), range 1–255
`reboot_times`	number		Maximum power-off reboot count, range 0–255

yruo_8021x (802.1X RADIUS authentication configuration)

Field	Type	Description
`ieee802_1x`	number	Enable 802.1X authentication: `0` / `1`
`auth_addr`	string	RADIUS server IPv4 address, required when `ieee802_1x=1`
`auth_port`	number	RADIUS server port, range 1–65535, default `1812`
`auth_secret`	string	RADIUS shared secret, length 1–127, only letters, digits, and some special characters allowed
`nas_id`	string	NAS identifier, max 127 characters

Speed enum (`speed` field)

Value	Description
`0`	auto (auto-negotiation)
`1`	10M
`2`	100M
`3`	1000M (UR7xx models only)

2.1.3.2 Save Port Configuration

All port settings, PoE settings, and 802.1X settings are submitted via a single SET request; the `values` array contains the changed entries for each type.

Request

```
{
  "core": "yruo_port",
  "function": "set",
  "values": [
    {
      "base": "yruo_port",
      "type": "yruo_port",
      "index": "<port_index>",
      "value": {
        "status": 1,
        "property": 1,
```

```
        "speed": 0,
        "duplex": 0
      }
    },
    {
      "base": "yruo_poe",
      "type": "yruo_poe",
      "index": "qwerty",
      "value": {
        "poe": [
          {
            "port": "LAN1/WAN",
            "power_enable": 1,
            "description": "",
            "priority": 0,
            "icmp_enable": 1,
            "icmp_dest": "8.8.8.8",
            "icmp_interval": 300,
            "icmp_retry_interval": 5,
            "icmp_timeout": 3,
            "icmp_times": 3,
            "power_cut_interval": 3,
            "reboot_times": 3
          }
          // ... remaining port entries (including unmodified ones)
        ]
      }
    },
    {
      "base": "yruo_8021x",
      "type": "yruo_8021x",
      "index": "1aAbB2cCdDeE3",
      "value": {
        "ieee802_1x": 1,
        "auth_addr": "192.168.1.100",
        "auth_port": 1812,
        "auth_secret": "secret123",
        "nas_id": ""
      }
    }
  ]
}
```

- `yruo_port` entries contain only the changed ports (per actual modifications); there is no need to submit all ports.
- `yruo_poe`'s `value.poe` must be submitted as the **full array** (including unmodified entries); the order is the `priority` sort result.
- `yruo_poe` and `yruo_8021x` are included in `values` only when the device supports the corresponding feature.

2.1.4 WAN

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request sequence number
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM code
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.1.4.1 Query WAN Configuration

> **Note**: On page load, `core: 'yruo_port'` is requested first to check whether the WAN port has been switched to LAN (see 2.1.3 Port interface). If it has been switched, the WAN configuration page is not loaded.

Request

```
{
  "core": "yruo_wan",
  "function": "get",
  "values": [{ "base": "yruo_wan" }]
}
```

Response


```
{
  "id": 46,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_wan",
          "index": "e74QlGe9uOXfGgOTT54i9B3kXFwV34A8Lz89M7Oas8Anv5cwO3C",
          "value": {
            "grey": 0,
            "enable_wan": 1,
```

```
            "port": "WAN/LAN5",
            "protocol": 0,
            "mtu": 1500,
            "enable_nat": 1,
            "gateway": "192.168.50.1",
            "pri_dns": "192.168.1.1",
            "sec_dns": "8.8.8.8",
            "pri_dns6": "",
            "sec_dns6": "",
            "static": {
              "ip_address": "192.168.50.3",
              "netmask": "255.255.255.0",
              "gateway": "192.168.50.1",
              "ip6addr": "2001:db8:85a3::8a2e:370:7334",
              "ip6prefixlen": 64,
              "ip6gw": "",
              "multi_ip": []
            },
            "dhcp": {
              "mtu": 1500,
              "use_peer_dns": 0
            },
            "pppoe": {
              "mtu": 1500,
              "use_peer_dns": 0
            },
            "pppoev6": {
              "mtu": 1500,
              "use_peer_dns": 0
            }
          }
        }
      ]
    }
  ]
}
```


> - Each WAN interface corresponds to one `yruo_wan` record; `index` is an opaque string that must be passed back unchanged on save.
> - The `static`, `dhcp`, `pppoe`, `pppoev6`, `dhcv6`, `ds`, and `6rd` sub-objects each store the settings for the corresponding protocol. Sub-objects for unconfigured protocols may not appear in the response.

Field Descriptions

Top-level fields

Field	Type	Read-only	Description
`grey`	number	✅	Whether the WAN interface has been disabled: `0`=normal, `1`=WAN port switched to LAN
`enable_wan`	number		Enable WAN interface: `0` / `1`

Field	Type	Read-only	Description
`port`	string	✅	Bound physical port name, e.g. `"WAN/LAN5"`
`protocol`	number		Access protocol, see protocol enum
`mtu`	number		MTU, range 68–1500
`enable_nat`	number		Enable NAT: `0` / `1`
`gateway`	string		Currently effective gateway (top-level, same as `static.gateway`)
`pri_dns`	string		Primary IPv4 DNS
`sec_dns`	string		Secondary IPv4 DNS
`pri_dns6`	string		Primary IPv6 DNS
`sec_dns6`	string		Secondary IPv6 DNS

static sub-object (protocol=0, Static IP)

Field	Type	Description
`ip_address`	string	IPv4 address, required
`netmask`	string	Netmask, required
`gateway`	string	IPv4 gateway, max 32 characters (IP or domain), required
`ip6addr`	string	IPv6 address
`ip6prefixlen`	number	IPv6 prefix length, range 1–128
`ip6gw`	string	IPv6 gateway
`multi_ip`	array	Additional IP address list, up to 10 entries, each containing `ip_addr` (IPv4) and `mask` (netmask)

dhcp sub-object (protocol=1, DHCP Client)

Field	Type	Description
`mtu`	number	MTU, range 68–1500
`use_peer_dns`	number	Use peer DNS: `0` =no (manual), `1` =yes; when `0`, `pri_dns` / `sec_dns` take effect

pppoe sub-object (protocol=2, PPPoE)

Field	Type	Description
`mtu`	number	MTU, range 68–1500

Field	Type	Description
`use_peer_dns`	number	Use peer DNS: `0`=no (manual), `1`=yes
`user`	string	PPPoE username, max 63 characters, required
`password`	string	PPPoE password, max 63 characters, required (not returned in GET response)
`service_name`	string	Service name, max 63 characters
`redial_interval`	number	Redial interval (seconds), range 1–600
`retries`	number	Maximum retry count, range 0–9

pppoev6 sub-object (protocol=5, PPPoEv6, hidden in current UI version)

Field	Type	Description
`mtu`	number	MTU, range 68–1500
`use_peer_dns`	number	Use peer DNS: `0` / `1`
`user`	string	PPPoEv6 username, max 63 characters
`password`	string	PPPoEv6 password (not returned in GET response)
`service_name`	string	Service name
`redial_interval`	number	Redial interval (seconds), range 1–600
`retries`	number	Maximum retry count, range 0–9

dhcv6 sub-object (protocol=3, DHCPv6 Client, not present in this device's response)

Field	Type	Description
`mtu`	number	MTU, range 68–1500
`request_mode`	number	IPv6 address request mode, see request_mode enum
`prefix_len`	number	Requested IPv6 prefix length, range 0–64

ds sub-object (protocol=6, Dual Stack Lite, not present in this device's response)

Field	Type	Description
`mtu`	number	MTU, range 68–1500
`aftr_addr`	string	AFTR address (IPv6 with prefix), max 64 characters
`local_addr`	string	Local IPv6 address (with prefix), max 64 characters
`b4_gate`	string	B4 IPv6 gateway

6rd sub-object (protocol=7, 6rd IPv6-over-IPv4, not present in this device's response)

Field	Type	Description
`mtu`	number	MTU, range 68–1500
`local_ipv4`	string	Local IPv4 address
`remote_ipv4`	string	Remote IPv4 address
`ipv6_prefix`	string	IPv6 prefix address
`ipv6_prefixlen`	number	IPv6 prefix length
`ipv4_prefixlen`	number	IPv4 prefix length
`b4_gate`	string	B4 IPv6 gateway

Protocol enum (`protocol` field)

Value	Description
`0`	Static IP
`1`	DHCP Client
`2`	PPPoE
`3`	DHCPv6 Client
`5`	PPPoEv6 (hidden in current UI version, firmware still supports it)
`6`	Dual Stack Lite (DS-Lite)
`7`	6rd (IPv6-over-IPv4, firmware supports it, option not exposed in UI)

request_mode enum (`dhcv6.request_mode` field, known enum values, may be incomplete)

Value	Description
`0`	Mode 0
`1`	Mode 1
`2`	Mode 2

2.1.4.2 Save WAN Configuration

Request

The SET request's `value` is a **flattened structure**: on page load, fields from each protocol sub-object (`static`, `dhcp`, `pppoe`, etc.) are merged into the top level; on save, top-level fields are submitted directly, without nesting sub-objects.

```
{
  "core": "yruo_wan",
  "function": "set",
  "values": [
    {
      "base": "yruo_wan",
      "type": "yruo_wan",
      "index": "e74QlGe9uOXfGgOTT54i9B3kXFwV34A8Lz89M7Oas8Anv5cwO3C",
      "value": {
        "enable_wan": 1,
        "protocol": 0,
        "mtu": 1500,
        "enable_nat": 1,
        "ip_address": "192.168.50.3",
        "netmask": "255.255.255.0",
        "gateway": "192.168.50.1",
        "ip6addr": "2001:db8:85a3::8a2e:370:7334",
        "ip6prefixlen": 64,
        "ip6gw": "",
        "multi_ip": [],
        "pri_dns": "192.168.1.1",
        "sec_dns": "8.8.8.8",
        "pri_dns6": "",
        "sec_dns6": ""
      }
    }
  ]
}
```


> - The example above is the save structure for `protocol=0` (Static IP); for other protocols, `value` contains the fields of the corresponding protocol.
> - `use_peer_dns` field: in DHCP/PPPoE/PPPoEv6 modes, if peer DNS is enabled, `pri_dns` / `sec_dns` need not be submitted.
> - PPPoE's `password` field: plaintext on SET, not returned in GET response.
> - `multi_ip` is the full array, including unmodified entries.

2.1.5 Bridge

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request sequence number
`model`	string	Device model
`pn`	string	Product number

Field	Type	Description
`oem`	string	OEM code
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.1.5.1 Query Bridge Configuration

Request

```
{
  "core": "yruo_bridge",
  "function": "get",
  "values": [{ "base": "yruo_bridge" }]
}
```

Response

```
{
  "id": 47,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_bridge",
          "index": 1,
          "value": {
            "stp": 0,
            "name": "Bridge0",
            "mtu": 1500,
            "ip_address": "192.168.1.1",
            "netmask": "255.255.255.0",
            "ipv6_address": "",
            "multi_ip": [],
            "deletable": [1, 1],
            "members": ["vlan 1", "WLAN"],
            "list": ["WLAN"],
            "ip_passthrough_enable": false
          }
        }
      ]
    }
  ]
}
```

> The `get` entry may also contain a top-level field `grey` (outside `value`): if truthy, the entire bridge configuration UI becomes read-only.

Field Descriptions

Editable fields

Field	Type	Description
`stp`	number	Enable STP (Spanning Tree Protocol): `0` / `1`
`ip_address`	string	IPv4 address, required; read-only when `ip_passthrough_enable=true`
`netmask`	string	Netmask, required; read-only when `ip_passthrough_enable=true`
`ipv6_address`	string	IPv6 address (including prefix length, e.g. `"2001:db8::1/64"`), may be an empty string
`mtu`	number	MTU, range 68–1500, required
`multi_ip`	array	Additional IP address list, up to 10 entries, each containing `ip_addr` (IPv4) and `mask` (netmask); read-only when `ip_passthrough_enable=true`

Read-only fields

Field	Type	Description
`name`	string	Bridge name, e.g. `"Bridge0"`
`ip_passthrough_enable`	boolean	Whether IP Passthrough mode is enabled; when `true`, IP address fields are all read-only
`members`	string[]	Current bridge member interface list, e.g. `["vlan1", "WLAN"]`
`deletable`	number[]	One-to-one correspondence with `members`, marking whether each member can be deleted: `0`=no, `1`=yes
`list`	string[]	List of interfaces available to be added to the bridge

---

2.1.5.2 Save Bridge Configuration

Request

```
{
  "core": "yruo_bridge",
  "function": "set",
  "values": [
    {
```

```
      "base": "yruo_bridge",
      "type": "yruo_bridge",
      "index": 1,
      "value": {
        "stp": 0,
        "ip_address": "192.168.1.1",
        "netmask": "255.255.255.0",
        "ipv6_address": "",
        "mtu": 1500,
        "multi_ip": []
      }
    }
  ]
}
```

> - `multi_ip` must be submitted as the **full array** (including unmodified entries).
> - `name`, `members`, `deletable`, `list`, and `ip_passthrough_enable` are read-only fields and need not be submitted on SET.

2.1.6 WLAN

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.
> The WPA Enterprise certificate upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with `multipart/form-data` format.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request sequence number
`model`	string	Device model, e.g. `"UR35"`
`pn`	string	Product number
`oem`	string	OEM code
`rtver`	string	Firmware version
`status`	number	Status code, `0` indicates success

2.1.6.1 Query

Request

```
{
  "core": "yruo_wifi",
  "function": "get",
  "values": [{ "base": "yruo_wifi" }]
}
```

Response


```
{
  "id": 48,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_wifi",
          "index": "1",
          "value": {
            "enabled": 1,
            "mode": 0,
            "country_code": "CN",
            "bssid": "24:e1:24:fe:b7:35",
            "hwmodes": 14,
            "hw_mode": 4,
            "channel": 0,
            "ht_capab": 0,
            "channel_list": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
            "channel_list1": [],
            "scan": 0,
            "bridged": 1,
            "ip": "192.168.1.1",
            "mask": "255.255.255.0",
            "multi_ssid": [
              {
                "id": "2jKuqOsr6ybOYZWb5HkG35O8U1zW1O6k5EY831sgx8PLsAMTHUrO",
                "ssid_enable": 1,
                "ssid": "Router_FEB735",
                "ssid_sta": "Router_FEB735",
                "desc": "",
                "encryption": 5,
                "cipher": 0,
                "key": "iotpassword",
                "key0": "",
                "broadcast_ssid": 1,
                "ap_isolate": 0,
                "guest_network": 0,
                "max_sta": 10
              }
            ],
            "filter_type": 0,
            "white_list": [],
            "black_list": [],
            "auth_addr": "",
            "auth_port": 1812,
            "auth_secret": "",
            "acct_addr": "",
            "acct_port": 1813,
```

```
            "acct_secret": "",
            "nas_id": ""
          }
        }
      ]
    }
  ]
}
```

> Single-radio devices (3X/4X) have only one record with `index: "1"`; dual-radio devices (7X) have two records, `index: "1"` and `index: "2"`.

Field Descriptions

yruo_wifi value top-level fields

Field	Type	Description
`enabled`	number	WLAN switch: `0`=disabled, `1`=enabled
`mode`	number	Operating mode, see mode enum
`country_code`	string	Country/region code, e.g. `"CN"` (the setting option is shown only when the URL contains a `?cc` parameter)
`bssid`	string	Local WLAN MAC address (read-only)
`hwmodes`	number	Bitmask of supported radio types, bit0=a(5GHz), bit1=b(2.4GHz), bit2=g(2.4GHz), bit3=n, bit4=ac(5GHz)
`hw_mode`	number	Current radio type, see hw_mode enum; available options are determined by `hwmodes`
`channel`	number	Channel, `0`=auto; selectable values come from `channel_list` (2.4GHz) or `channel_list1` (5GHz/ac)
`channel_list`	array<number>	Available 2.4GHz channel list (for hw_mode=1/2/4)
`channel_list1`	array<number>	Available 5GHz channel list (for hw_mode=0/5/6)
`ht_capab`	number	Channel bandwidth: `0`=20MHz, `1`=40MHz, `2`=80MHz; locked to 0 for 802.11b/g, locked to 2 for 802.11ac/a
`scan`	number	Reserved field (used by the UI scan button)
`bridged`	number	Whether bridged to LAN: `0`=no, `1`=yes (read-only)
`ip`	string	Local IP address (settable in AP mode when the guest network is enabled; settable in Client mode + Static IP)
`mask`	string	Netmask

Field	Type	Description
`proto`	number	IP protocol (only present in Client mode or when the guest network is enabled): `1`=Static IP, `2`=DHCP Client
`gate`	string	Default gateway (only present in Client mode + Static IP)
`filter_type`	number	MAC address filter type, see filter_type enum
`white_list`	array<object>	MAC whitelist list, see white_list/black_list entry descriptions; takes effect when `filter_type=1`
`black_list`	array<object>	MAC blacklist list; takes effect when `filter_type=2`
`auth_addr`	string	RADIUS authentication server address (used in AP mode WPA-EAP)
`auth_port`	number	RADIUS authentication server port, default 1812
`auth_secret`	string	RADIUS authentication secret
`acct_addr`	string	RADIUS accounting server address (optional; required together with acct_port/acct_secret when present)
`acct_port`	number	RADIUS accounting server port, default 1813
`acct_secret`	string	RADIUS accounting secret
`nas_id`	string	NAS identifier

multi_ssid[] entry fields

Field	Type	Description
`id`	string	SSID unique identifier (read-only, must be passed back on save)
`ssid_enable`	number	SSID enabled state: `0`=disabled, `1`=enabled
`ssid`	string	SSID name (shown/set in AP mode)
`ssid_sta`	string	Target AP SSID (shown/set in Client mode)
`desc`	string	Description
`encryption`	number	Encryption method, see encryption enum
`cipher`	number	Encryption algorithm: `0`=Auto, `1`=AES, `2`=TKIP, `3`=AES/TKIP
`key`	string	WPA/WPA2 password (PSK mode, 8-63 ASCII characters)
`key0`	string	WEP key (WEP mode; ASCII 5/13 chars or HEX 10/26 chars)

Field	Type	Description
`broadcast_ssid`	number	SSID broadcast: `0`=hidden, `1`=broadcast
`ap_isolate`	number	AP isolation: `0`=off, `1`=on
`guest_network`	number	Guest network: `0`=off, `1`=on; when on, AP mode uses a separate IP subnet
`max_sta`	number	Maximum associated clients, range 1-15

white_list / black_list entry fields

Field	Type	Description
`mac`	string	MAC address, format `xx:xx:xx:xx:xx:xx`
`desc`	string	Remark description, 1-63 characters

mode enum

Value	Meaning
`0`	AP (Access Point)
`1`	Client (connects to an external AP)

hw_mode enum

Value	Meaning
`0`	802.11a (5GHz)
`1`	802.11b (2.4GHz)
`2`	802.11g (2.4GHz)
`4`	802.11n (2.4GHz)
`5`	802.11ac (5GHz)
`6`	802.11n (5GHz)

> The `hwmodes` bitmask determines which hw_mode values the device actually supports. In the example response `hwmodes=14` (bit1+bit2+bit3) = supports b/g/n (2.4GHz).

encryption enum

Value	Meaning	Applicable modes
`0`	No Encryption	AP/Client
`1`	WEP Open System/Shared Key	AP/Client
`3`	WPA-PSK	AP/Client

Value	Meaning	Applicable modes
4	WPA2-PSK	AP/Client
5	WPA-PSK/WPA2-PSK	AP/Client
6	WPA-EAP (AP mode) / WPA-Enterprise (Client mode)	AP/Client
7	WPA2-EAP (AP mode) / WPA2-Enterprise (Client mode)	AP/Client
8	WPA-Enterprise/WPA2-Enterprise	Client only

filter_type enum

Value	Meaning
0	Disable MAC filtering
1	Whitelist (allow only MACs in white_list)
2	Blacklist (block MACs in black_list)

2.1.6.2 Save

Request


```
{
  "core": "yruo_wifi",
  "function": "set",
  "values": [
    {
      "base": "yruo_wifi",
      "type": "yruo_wifi",
      "index": "1",
      "value": {
        "enabled": 1,
        "mode": 0,
        "hw_mode": 4,
        "channel": 6,
        "ht_capab": 1,
        "filter_type": 0,
        "white_list": [],
        "black_list": [],
        "auth_addr": "",
        "auth_port": 1812,
        "auth_secret": "",
        "acct_addr": "",
        "acct_port": 1813,
        "acct_secret": "",
        "nas_id": "",
        "multi_ssid": [
          {
            "id": "2jKuqOsr6ybOYZWb5HkG35O8U1zW1O6k5EY831sgx8PLsAMTHUrO",
            "action": 2,
```

```
            "ssid": "MyWifi",
            "encryption": 5,
            "cipher": 0,
            "key": "mypassword",
            "broadcast_ssid": 1,
            "ap_isolate": 0,
            "guest_network": 0,
            "max_sta": 10
          }
        ]
      }
    }
  ]
}
```

Field Descriptions

AP mode save structure

The front-end `modifyValue` function extracts `ssid`, `encryption`, `key`, `key0`, `broadcast_ssid`, `ap_isolate`, `cipher`, `guest_network`, and `max_sta` from the top-level value and packages them into `multi_ssid[0]`; the remaining fields stay in the top-level value.

Additional `multi_ssid[0]` fields:

Field	Type	Description
`id`	string	SSID unique identifier, obtained from the GET response; for new entries, use a random hex string
`action`	number	`1`=mode changed, `2`=mode unchanged

Client mode save structure

In Client mode (`mode=1`), `multi_ssid` is not packaged; all fields are placed directly in the top-level value, including:

Field	Type	Description
`ssid_sta`	string	Target AP SSID
`bssid_sta`	string	Target AP BSSID (optional; at least one of ssid_sta/bssid_sta must be filled)
`encryption`	number	Encryption method (same enum as AP mode)
`cipher`	number	Encryption algorithm
`key`	string	WPA password (PSK mode)
`key0`	string	WEP key
`proto`	number	IP protocol: `1`=Static IP, `2`=DHCP Client
`ip`	string	Static IP (when proto=1)

Field	Type	Description
`mask`	string	Netmask (when proto=1)
`gate`	string	Default gateway (when proto=1)
`xsupplicant_type`	number	EAP type: `0`=Peap, `1`=Leap, `2`=TLS, `3`=TTLS (for Enterprise encryption)
`identity`	string	EAP identity (for Enterprise encryption)
`user`	string	EAP username (for Peap/TTLS types)
`key_eap`	string	EAP password (for non-TLS types, SET only)
`phase1`	string	EAP Phase1 parameters
`phase2`	string	EAP Phase2 parameters

> Client mode WPA Enterprise also requires uploading certificate files, see 2.1.6.4.

2.1.6.3 Wi-Fi Scan

Triggered by the scan button; retrieves a list of visible nearby APs.

Request

```
{
  "core": "yruo_scan",
  "function": "get",
  "values": [
    {
      "base": "yruo_scan",
      "index": "1"
    }
  ]
}
```

> `index` corresponds to the WLAN interface number (`"1"` or `"2"`).

Response field descriptions (yruo_scan)

Field	Type	Description
`ssid`	string	AP name
`channel`	number	Channel, `0`=unknown
`signal`	number	Signal strength (dBm)
`cipher`	number	Encryption algorithm: `0`=Auto, `1`=AES, `2`=TKIP, `3`=AES/TKIP
`bss`	string	AP BSSID (MAC address)
`auth`	number	Authentication/encryption method, enum same as encryption (0-8)

Field	Type	Description
`freq`	number	Frequency (MHz)

2.1.6.4 WPA Enterprise Certificate Upload

When choosing WPA-Enterprise / WPA2-Enterprise encryption in Client mode, the following certificate files must be uploaded.

Request

`POST https://<device-ip>/cgi-bin/file-import`, format `multipart/form-data`.

Form field	Description
`sessionid`	Session credential (same as the `x-session-id` value)
`filename`	Format `type=wpa_enterprise&file=<certificate type>`
`size`	File size (bytes)
`file`	File binary content

Uploadable certificate types (`file` field values):

file value	Description
`CA Certificate`	CA root certificate (required for Peap/Leap/TLS/TTLS types)
`Client Certificate`	Client certificate (required for TLS type)
`Client Private Key`	Client private key (required for TLS type)

Success response

```
{ "size": 1234, "checksum": "abc123", "filename": "WPA_ENTERPRISE_CA.crt" }
```

Failure response

```
{ "templet_code": 1, "params_count": 0, "templet_params": [] }
```

2.1.7 VLAN

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request sequence number
`model`	string	Device model, e.g. `"UR35"`
`pn`	string	Product number
`oem`	string	OEM code
`rtver`	string	Firmware version
`status`	number	Status code, `0` indicates success

2.1.7.1 Query

This page issues two sequential requests on load:

1. First query port status (`yruo_port`) to determine whether the WAN/LAN5 port has been switched to LAN mode;
2. Then query VLAN data (`yruo_lan`) for rendering, and decide based on the first step whether to show the port4 (WAN/LAN5) column.

Request 1: Query port status

```
{
  "core": "yruo_port",
  "function": "get",
  "values": [{ "base": "yruo_port" }]
}
```

The response structure and field meanings are identical to 2.1.3 Port; here it is used only to determine port properties and is not separately displayed or saved on this page.

Request 2: Query VLAN data

```
{
  "core": "yruo_lan",
  "function": "get",
  "values": [{ "base": "yruo_lan" }]
}
```

Response

```
{
  "id": 50,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
```

```
{
  "get": [
    {
      "type": "yruo_lan",
      "index": 0,
      "value": {
        "lan": [
          {
            "id": "nDO2xQyB",
            "name": "vlan1",
            "vid": 1,
            "ip": "192.168.1.1",
            "mask": "255.255.255.0",
            "addr_v6": "",
            "mtu": 1500,
            "allow_del": 0
          },
          {
            "id": "9bUG3o",
            "name": "vlan4",
            "vid": 100,
            "ip": "192.168.4.1",
            "mask": "255.255.255.0",
            "addr_v6": "",
            "mtu": 1500,
            "allow_del": 1
          }
          // ... remaining lan entries omitted
        ],
        "vlan": [
          {
            "id": "M6vhW5X",
            "vid": 1,
            "port0": 2,
            "port1": 2,
            "port2": 2,
            "port3": 2,
            "port4": 3,
            "cpu": 1,
            "allow_del": 0
          },
          {
            "id": "kA79u",
            "vid": 100,
            "port0": 1,
            "port1": 1,
            "port2": 1,
            "port3": 1,
            "port4": 3,
            "cpu": 1,
            "allow_del": 1
          }
          // ... remaining vlan entries omitted
        ],
        "ports": 15,
        "vid_wan": "4094",
```

```
            "tag_wan": "2"
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

yruo_lan value top-level fields

Field	Type	Description
`lan`	array<object>	LAN interface list, see lan[] entry fields
`vlan`	array<object>	VLAN rule list, see vlan[] entry fields
`ports`	number	Physical port availability bitmask: bit0=port0(LAN1), bit1=port1(LAN2), bit2=port2(LAN3), bit3=port3(LAN4); `15` (0b1111)=all available
`vid_wan`	string	WAN custom VLAN ID, range 2-4094 (read-only when the WAN port is switched to LAN)
`tag_wan`	string	WAN VLAN tag mode, see port enum; read-only when the WAN port is switched to LAN

lan[] entry fields

Field	Type	Description
`id`	string	Entry unique identifier (must be passed back on save)
`name`	string	Interface name (e.g. `"vlan1"`), max 20 characters; read-only when `allow_del=0`
`vid`	number	Bound VLAN ID; read-only when `allow_del=0`
`ip`	string	IP address; read-only when `vid=1`
`mask`	string	Netmask; read-only when `vid=1`
`addr_v6`	string	IPv6 address (required together with `prefix_v6` when configured)
`prefix_v6`	number	IPv6 prefix length, range 0-128 (present when `addr_v6` is configured)
`mtu`	number	MTU, range 68-1500; read-only when `vid=1`
`allow_del`	number	Whether deletion is allowed: `0`=not deletable, `1`=deletable

vlan[] entry fields

Field	Type	Description
`id`	string	Entry unique identifier (must be passed back on save)
`vid`	number	VLAN ID; the first entry is 1 (read-only), others range 2-4093
`port0`	number	LAN1 port mode, see port enum (on dual-port models this is the LAN1/WAN port)
`port1`	number	LAN2 port mode
`port2`	number	LAN3 port mode (only on four-port and above models)
`port3`	number	LAN4 port mode (only on four-port and above models)
`port4`	number	WAN/LAN5 port mode (this column appears only when the WAN port is switched to LAN)
`cpu`	number	CPU port mode, see port enum
`allow_del`	number	Whether deletion is allowed: `0`=not deletable, `1`=deletable

> Each physical port may have at most one Untagged record across all VLANs; port columns whose bit is 0 in the `ports` bitmask are not editable.

port enum (applies to port0-port4, cpu, tag_wan)

Value	Meaning
`1`	Tagged
`2`	Untagged
`3`	Close

2.1.7.2 Save

Request

```
{
  "core": "yruo_lan",
  "function": "set",
  "values": [
    {
      "base": "yruo_lan",
      "type": "yruo_lan",
      "index": 0,
      "value": {
        "lan": [
          {
            "id": "nDO2xQyB",
            "name": "vlan1",
            "vid": 1,
```

```
            "ip": "192.168.1.1",
            "mask": "255.255.255.0",
            "addr_v6": "",
            "mtu": 1500,
            "allow_del": 0
          }
          // ... full lan array
        ],
        "vlan": [
          {
            "id": "M6vhW5X",
            "vid": 1,
            "port0": 2,
            "port1": 2,
            "port2": 2,
            "port3": 2,
            "port4": 3,
            "cpu": 1,
            "allow_del": 0
          }
          // ... full vlan array
        ],
        "vid_wan": "4094",
        "tag_wan": "2"
      }
    }
  ]
}
```

Notes

- SET submits the full `lan[]` and `vlan[]` arrays (including unmodified entries).
- New entries have no `id` field (or the front end generates a random string); when deleting an entry, remove it from the array.
- `vid_wan` and `tag_wan` are string types (consistent with the GET response).
- The `allow_del` field must be passed back together with the entry.

2.1.8 Loopback

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request sequence number
`model`	string	Device model, e.g. `"UR35"`
`pn`	string	Product number

Field	Type	Description
`oem`	string	OEM code
`rtver`	string	Firmware version
`status`	number	Status code, `0` indicates success

2.1.8.1 Query

Request

```
{
  "core": "yruo_loopback",
  "function": "get",
  "values": [{ "base": "yruo_loopback" }]
}
```

Response

```
{
  "id": 51,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_loopback",
          "index": "qwerty",
          "value": {
            "lo_ip": "127.0.0.1",
            "lo_mask": "255.0.0.0"
          }
        }
      ]
    }
  ]
}
```

> If additional IPs are configured, `get[]` contains extra `yruo_loopback` entries, each containing `ip_addr` and `mask` fields (see the field descriptions below). Up to 10 additional IPs.

Field Descriptions

Main entry (lo_ip / lo_mask)

Field	Type	Description
`lo_ip`	string	Loopback interface IP address, fixed at `127.0.0.1` (read-only)
`lo_mask`	string	Loopback interface netmask, fixed at `255.0.0.0` (read-only)

Additional IP entry (ip_addr / mask)

When additional IPs exist, an extra `yruo_loopback` entry appears in `get[]`:

Field	Type	Description
`ip_addr`	string	Additional IP address
`mask`	string	Additional IP netmask, default `255.255.255.255`

2.1.8.2 Save

Request

```
{
  "core": "yruo_loopback",
  "function": "set",
  "values": [
    {
      "base": "yruo_loopback",
      "type": "yruo_loopback",
      "index": "qwerty",
      "value": {
        "ip_addr": "10.0.0.1",
        "mask": "255.255.255.255"
      }
    }
  ]
}
```

Notes

- `lo_ip` and `lo_mask` are read-only fields and need not be submitted.
- Additional IP entries are submitted via the `ip_addr` and `mask` fields, using the same `index` as the main entry.
- When adding a new additional IP, there is no `index` (or use a random string); when deleting, remove the corresponding entry from the `values` array.

2.2 DHCP

2.2.1 DHCP Server

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

2.2.1.1 Query DHCP Server Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_dhcpserver",
  "values": [
    { "base": "yruo_dhcpserver" }
  ]
}
```

Response

`result[0].get[]` each item corresponds to the DHCP server configuration of one network interface; `index` is the interface name.

```
{
  "id": 53,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_dhcpserver",
```

```
          "index": "Bridge0",
          "value": {
            "enable": true,
            "ip_passthrough_enable": false,
            "relay_enabled": false,
            "network": "Bridge0",
            "ifmask": "255.255.255.0",
            "ifaddr": "192.168.1.1",
            "network_array": ["WAN/LAN5", "vlan4", "vlan3", "vlan2", "Bridge0"],
            "start_ip": "192.168.1.100",
            "end_ip": "192.168.1.199",
            "mask": "255.255.255.0",
            "lease": 1440,
            "wins": "",
            "pri_dns": "192.168.1.1",
            "sec_dns": "8.8.8.8",
            "static_ip_map": []
          }
        },
        {
          "type": "yruo_dhcpserver",
          "index": "vlan2",
          "value": {
            "enable": false,
            "ip_passthrough_enable": false,
            "relay_enabled": false,
            "network": "vlan2",
            "ifmask": "255.255.255.0",
            "ifaddr": "192.168.2.1",
            "network_array": ["WAN/LAN5", "vlan4", "vlan3", "vlan2", "Bridge0"],
            "start_ip": "192.168.2.5",
            "end_ip": "192.168.2.55",
            "mask": "255.255.255.0",
            "lease": 720,
            "wins": "",
            "pri_dns": "192.168.2.5",
            "sec_dns": "",
            "static_ip_map": []
          }
        }
        // ... more interface entries (the actual response includes all
configured interfaces)
      ]
    }
  ]
}
```


Field Descriptions

`yruo_dhcpserver` **type (type=**`yruo_dhcpserver`**, index=interface name string)**

Field	Type	Description
`enable`	boolean	DHCP server enable; cannot be turned on when DHCP Relay is enabled

Field	Type	Description
`ip_passthrough_enable`	boolean	Whether IP Passthrough is enabled (read-only); when `true`, all configuration items of the Bridge0 interface cannot be modified
`relay_enabled`	boolean	Whether DHCP Relay is enabled (read-only); relay and server are mutually exclusive
`network`	string	Bound network interface, options come from `network_array`; an enabled interface cannot be bound twice
`network_array`	array<string>	List of selectable network interfaces (read-only), shared by all entries
`ifaddr`	string	Interface current IP address (read-only), conditional field (returned only when the interface has an IP configured)
`ifmask`	string	Interface current netmask (read-only), conditional field (returned only when the interface has an IP configured)
`start_ip`	string	Address pool start IP (IPv4)
`end_ip`	string	Address pool end IP (IPv4); must be greater than `start_ip` and in the same subnet as `start_ip`
`mask`	string	Netmask
`lease`	integer	Lease time (minutes), range 5–1440
`pri_dns`	string	Primary DNS server (IPv4)
`sec_dns`	string	Secondary DNS server (IPv4), may be empty
`wins`	string	WINS server (IPv4), may be empty
`static_ip_map`	array	MAC-IP static binding list, up to 100 entries, see table below

`static_ip_map` entry fields

Field	Type	Description
`hw_addr`	string	MAC address, format `xx:xx:xx:xx:xx:xx`
`ip`	string	Bound static IP (IPv4); must be in the same subnet as the address pool but cannot fall within the `start_ip`–`end_ip` range

2.2.1.2 Save DHCP Server Configuration

Request

The `values` array contains only the interface entries that changed.

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_dhcpserver",
  "values": [
    {
      "base": "yruo_dhcpserver",
      "type": "yruo_dhcpserver",
      "index": "Bridge0",
      "value": {
        "enable": true,
        "network": "Bridge0",
        "start_ip": "192.168.1.100",
        "end_ip": "192.168.1.199",
        "mask": "255.255.255.0",
        "lease": 1440,
        "pri_dns": "192.168.1.1",
        "sec_dns": "8.8.8.8",
        "wins": "",
        "static_ip_map": [
          { "hw_addr": "AA:BB:CC:DD:EE:FF", "ip": "192.168.1.50" }
        ]
      }
    }
  ]
}
```

`ip_passthrough_enable`, `relay_enabled`, `network_array`, `ifaddr`, and `ifmask` are read-only fields and cannot be submitted.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.2.2 DHCPv6 Server

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

2.2.2.1 Query DHCPv6 Server Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_dhcpv6server",
  "values": [
    { "base": "yruo_dhcpv6server" }
  ]
}
```

Response

`result[0].get[]` each item corresponds to the DHCPv6 server configuration of one network interface; `index` is the interface name or a random ID (when no interface is bound).

```
{
  "id": 54,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_dhcpv6server",
          "index": "Bridge0",
          "value": {
```

```
            "enable": true,
            "ip_passthrough_enable": false,
            "relay_enabled": false,
            "network": "Bridge0",
            "network_array": ["WAN/LAN5", "vlan4", "vlan3", "vlan2", "Bridge0"],
            "start_ipv6": "2001:DOB0:3000:3001::100",
            "end_ipv6": "2001:DOB0:3000:3001::199",
            "mask": "64",
            "lease": 1440,
            "pri_dns": "2001:DOB0:3000:3001::1",
            "sec_dns": "2001:4860:4860::8888",
            "static_ipv6_map": []
          }
        },
        {
          "type": "yruo_dhcpv6server",
          "index": "rgzcnKuIj",
          "value": {
            "enable": 0,
            "ip_passthrough_enable": false,
            "relay_enabled": false,
            "network": "",
            "network_array": ["WAN/LAN5", "vlan4", "vlan3", "vlan2", "Bridge0"],
            "start_ipv6": "",
            "end_ipv6": "",
            "mask": ""
          }
        }
        // ... more entries (the actual response includes all configured slots)
      ]
    }
  ]
}
```


> The `enable` field of a configured interface (e.g. `index`=`"Bridge0"`) is boolean; the `enable` field of an empty entry for an unbound interface is integer `0`. `lease`, `pri_dns`, `sec_dns`, and `static_ipv6_map` are returned only in entries where `enable`=`true` and the interface is configured.

Field Descriptions

`yruo_dhcpv6server` type (type=`yruo_dhcpv6server`, index=interface name string or random ID)

Field	Type	Description
`enable`	boolean / integer	DHCPv6 server enable; boolean for configured entries, integer `0` for empty entries
`ip_passthrough_enable`	boolean	Whether IP Passthrough is enabled (read-only); when `true`, the Bridge0 interface configuration cannot be modified

Field	Type	Description
`relay_enabled`	boolean	Whether DHCP Relay is enabled (read-only); relay and server are mutually exclusive
`network`	string	Bound network interface, options come from `network_array`; empty string means unbound
`network_array`	array<string>	List of selectable network interfaces (read-only), shared by all entries
`start_ipv6`	string	Address pool start IPv6 address
`end_ipv6`	string	Address pool end IPv6 address
`mask`	string	IPv6 prefix length (numeric string, e.g. `"64"`)
`lease`	integer	Lease time (minutes), range 5–1440; present when `enable` = `true`
`pri_dns`	string	Primary DNS server (IPv6); present when `enable` = `true`
`sec_dns`	string	Secondary DNS server (IPv6), may be empty; present when `enable` = `true`
`static_ipv6_map`	array	DUID-IPv6 static binding list, up to 100 entries; present when `enable` = `true`, see table below

`static_ipv6_map` entry fields

Field	Type	Description
`hw_addr`	string	Client DUID (or MAC address)
`ipv6`	string	Bound static IPv6 address

2.2.2.2 Save DHCPv6 Server Configuration

Request

The `values` array contains only the interface entries that changed.

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_dhcpv6server",
  "values": [
    {
      "base": "yruo_dhcpv6server",
      "type": "yruo_dhcpv6server",
      "index": "Bridge0",
```

```
      "value": {
        "enable": true,
        "network": "Bridge0",
        "start_ipv6": "2001:DOB0:3000:3001::100",
        "end_ipv6": "2001:DOB0:3000:3001::199",
        "mask": "64",
        "lease": 1440,
        "pri_dns": "2001:DOB0:3000:3001::1",
        "sec_dns": "",
        "static_ipv6_map": []
      }
    }
  ]
}
```

`ip_passthrough_enable`, `relay_enabled`, and `network_array` are read-only fields and cannot be submitted.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.2.3 DHCP Relay

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

2.2.3.1 Query DHCP Relay Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_dhcprelay",
  "values": [
    { "base": "yruo_dhcprelay" }
  ]
}
```

Response

```
{
  "id": 55,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_dhcprelay",
          "index": 0,
          "value": {
            "enable": false,
            "server_enabled": true
          }
        }
      ]
    }
  ]
}
```

> `help_address` (the relay target DHCP server address) does not appear in the actual response; inferred to be not returned when relay is disabled.

Field Descriptions

`yruo_dhcprelay` **type (type=**`yruo_dhcprelay`**, index=**`0`**)**

Field	Type	Description
`enable`	boolean	DHCP Relay enable; cannot be turned on when the DHCP server is enabled (mutually exclusive)
`server_enabled`	boolean	Whether the DHCP server is enabled (read-only); when `true`, relay cannot be enabled

Field	Type	Description
`help_address`	string	Target DHCP server address, may contain multiple IPv4 addresses (comma-separated), max 127 characters; returned when `enable`=`true` (inferred from code; this field is not returned in the actual disabled state)

2.2.3.2 Save DHCP Relay Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_dhcprelay",
  "values": [
    {
      "base": "yruo_dhcprelay",
      "type": "yruo_dhcprelay",
      "index": 0,
      "value": {
        "enable": true,
        "help_address": "10.0.0.1,10.0.0.2"
      }
    }
  ]
}
```

`server_enabled` is a read-only field and cannot be submitted.

Response

Success: `status`=`0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.3 Firewall

2.3.1 Security

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

2.3.1.1 Query Security Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_firewall_security",
  "values": [
    { "base": "yruo_firewall_security" }
  ]
}
```

Response

```
{
  "id": 6,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_firewall_security",
          "index": 1,
          "value": {
            "http_port": 80,
            "http_remote": 1,
            "http_local": 1,
            "https_port": 443,
            "https_remote": 1,
            "https_local": 1,
            "telnet_port": 23,
            "telnet_remote": 1,
            "telnet_local": 1,
```

```
            "ssh_port": 22,
            "ssh_remote": 1,
            "ssh_local": 1,
            "ftp_port": 21,
            "ftp_remote": 0,
            "ftp_local": 0,
            "python_port": 9001,
            "python_remote": 1,
            "python_local": 1,
            "protect_ddos": 1,
            "url": [],
            "keyword": [],
            "redirect": 1
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_firewall_security` **type (type=**`yruo_firewall_security`**, index=**`1`**)**

DDoS protection and redirection

Field	Type	Description
`protect_ddos`	integer	DDoS protection enable, `0`=disabled, `1`=enabled
`redirect`	integer	Redirect HTTP to HTTPS, `0`=disabled, `1`=enabled; when enabled, HTTP and HTTPS local/remote access must both be on or both off

Access service control (each service includes three fields: port, local access, remote access)

Field	Type	Description
`http_port`	integer	HTTP service port, range 1–65535
`http_local`	integer	HTTP local access, `0`=disabled, `1`=enabled
`http_remote`	integer	HTTP remote access, `0`=disabled, `1`=enabled
`https_port`	integer	HTTPS service port, range 1–65535
`https_local`	integer	HTTPS local access, `0`=disabled, `1`=enabled
`https_remote`	integer	HTTPS remote access, `0`=disabled, `1`=enabled
`telnet_port`	integer	Telnet service port, range 1–65535
`telnet_local`	integer	Telnet local access, `0`=disabled, `1`=enabled
`telnet_remote`	integer	Telnet remote access, `0`=disabled, `1`=enabled

Field	Type	Description
`ssh_port`	integer	SSH service port, range 1–65535
`ssh_local`	integer	SSH local access, `0`=disabled, `1`=enabled
`ssh_remote`	integer	SSH remote access, `0`=disabled, `1`=enabled
`ftp_port`	integer	FTP service port, range 1–65535
`ftp_local`	integer	FTP local access, `0`=disabled, `1`=enabled
`ftp_remote`	integer	FTP remote access, `0`=disabled, `1`=enabled
`python_port`	integer	Python service port, range 1–65535
`python_local`	integer	Python local access, `0`=disabled, `1`=enabled
`python_remote`	integer	Python remote access, `0`=disabled, `1`=enabled

Website blocking

Field	Type	Description
`url`	array<string>	URL address block list, up to 10 entries, each max 127 characters
`keyword`	array<string>	Keyword block list, up to 10 entries, each max 64 characters

---

2.3.1.2 Save Security Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_firewall_security",
  "values": [
    {
      "base": "yruo_firewall_security",
      "type": "yruo_firewall_security",
      "index": 1,
      "value": {
        "protect_ddos": 1,
        "redirect": 0,
        "http_port": 80,
        "http_local": 1,
        "http_remote": 0,
        "https_port": 443,
        "https_local": 1,
        "https_remote": 1,
        "telnet_port": 23,
        "telnet_local": 0,
        "telnet_remote": 0,
        "ssh_port": 22,
```

```
        "ssh_local": 1,
        "ssh_remote": 0,
        "ftp_port": 21,
        "ftp_local": 0,
        "ftp_remote": 0,
        "python_port": 9001,
        "python_local": 1,
        "python_remote": 1,
        "url": ["http://example.com"],
        "keyword": ["ads"]
      }
    }
  ]
}
```

The `value` object only needs to include the changed fields.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.3.2 ACL

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

2.3.2.1 Query ACL Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_firewall_acl",
  "values": [
    { "base": "yruo_firewall_acl" }
  ]
}
```

Response

`result[0].get[]` contains entries of three types: `yruo_firewall_policy` (global policy, 1 entry), `yruo_firewall_acl` (ACL rules, 0 to many), `yruo_firewall_list` (interface bindings, 0 to many).

```
{
  "id": 7,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_firewall_policy",
          "index": 1,
          "value": {
            "policy": 1,
            "list": ["Bridge0", "PPPoE", "WAN/LAN5", "SIM1-APN1", "vlan2",
"vlan3", "vlan4", "tunl0", "gre0", "dmvpn"]
          }
        }
        // ... yruo_firewall_acl and yruo_firewall_list entries (empty when there
are no rules)
      ]
    }
  ]
}
```

Field Descriptions

`yruo_firewall_policy` **type (type=**`yruo_firewall_policy`**, index=**`1`**)**

Field	Type	Description
`policy`	integer	Default policy, `1`=Accept, `2`=Drop
`list`	array<string>	List of available network interfaces (read-only), used as dropdown options for interface binding

`yruo_firewall_acl` **type (type=`yruo_firewall_acl`, index=rule number)**

Field	Type	Description
`type`	integer	ACL type, `1`=extended (ID 100–199), `2`=standard (ID 1–99)
`id`	integer	ACL number; extended range 100–199, standard range 1–99
`action`	integer	Action, `1`=permit, `2`=deny
`protocol`	integer	Protocol, `0`=ip, `1`=icmp, `6`=tcp, `17`=udp, other values=custom protocol number (1–255); only valid for extended type
`source`	string	Source IP (IPv4), may be empty (meaning any)
`source_wildcard`	string	Source wildcard mask (inverse mask), e.g. `0.0.0.0`; required when `source` is non-empty
`destination`	string	Destination IP (IPv4), may be empty (meaning any); only valid for extended type
`destination_wildcard`	string	Destination wildcard mask; required when `destination` is non-empty; only valid for extended type
`s_port_type`	string	Source port match mode, `any` / `=` / `!=` / `<` / `>` / `between`; valid when protocol=tcp/udp
`s_port_s`	integer	Source port start value (or single port), range 1–65535; valid when `s_port_type` is not `any`
`s_port_e`	integer	Source port end value, range 1–65535; valid when `s_port_type`=`between`
`d_port_type`	string	Destination port match mode, same as `s_port_type`; valid when protocol=tcp/udp
`d_port_s`	integer	Destination port start value (or single port); valid when `d_port_type` is not `any`
`d_port_e`	integer	Destination port end value; valid when `d_port_type`=`between`
`icmp_type`	integer	ICMP type, range 0–255; valid when protocol=icmp
`icmp_code`	integer	ICMP code, range 0–255; valid when protocol=icmp
`more`	string	Additional rule parameters, max 255 characters; optional
`description`	string	Description, max 64 characters; optional

> Up to 128 ACL rules.

`yruo_firewall_list` **type (type=`yruo_firewall_list`, index=interface name string)**

Field	Type	Description
`interface`	string	Bound network interface, value taken from `yruo_firewall_policy.list`
`in_acl`	integer	Inbound ACL number, `-1` means not bound
`out_acl`	integer	Outbound ACL number, `-1` means not bound; `in_acl` and `out_acl` cannot both be `-1`

> Up to 128 interface bindings.

2.3.2.2 Save ACL Configuration

Request

The `values` array contains all changed entries and may include all three types simultaneously: `yruo_firewall_policy`, `yruo_firewall_acl`, and `yruo_firewall_list`.

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_firewall_acl",
  "values": [
    {
      "base": "yruo_firewall_acl",
      "type": "yruo_firewall_policy",
      "index": 1,
      "value": {
        "policy": 2
      }
    },
    {
      "base": "yruo_firewall_acl",
      "type": "yruo_firewall_acl",
      "index": 1,
      "value": {
        "type": 1,
        "id": 100,
        "action": 2,
        "protocol": 6,
        "source": "192.168.1.0",
        "source_wildcard": "0.0.0.255",
        "destination": "",
        "destination_wildcard": "0.0.0.0",
        "s_port_type": "any",
        "d_port_type": "=",
        "d_port_s": 80,
        "description": "block-http"
      }
    },
    {
```

```
      "base": "yruo_firewall_acl",
      "type": "yruo_firewall_list",
      "index": "WAN/LAN5",
      "value": {
        "interface": "WAN/LAN5",
        "in_acl": 100,
        "out_acl": -1
      }
    }
  ]
}
```

`yruo_firewall_policy.list` is a read-only field and cannot be submitted.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.3.3 Port Mapping

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

2.3.3.1 Query Port Mapping Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_firewall_port_mapping",
  "values": [
    { "base": "yruo_firewall_port_mapping" }
  ]
}
```

Response

`result[0].get[]` each item corresponds to one port mapping rule; `index` is the rule number.

```
{
  "id": 10,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_firewall_port_mapping",
          "index": 1,
          "value": {
            "src": "0.0.0.0/0",
            "to_dst": "2.2.2.2",
            "dport": "33",
            "to_dport": "22",
            "protocol": 1,
            "description": "123",
            "index": 1
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_firewall_port_mapping` **type (type=**`yruo_firewall_port_mapping`**, index=rule number)**

Field	Type	Description
`src`	string	Source IP/mask, format like `0.0.0.0/0`, max 18 characters

Field	Type	Description
`dport`	string	External destination port, supports a single port or a port range (e.g. `"80"` or `"80:90"`), max 11 characters
`to_dst`	string	Forwarding target IP/mask, max 18 characters
`to_dport`	string	Forwarding target port, supports a single port or a port range, max 11 characters
`protocol`	integer	Protocol, `1`=TCP, `2`=UDP, `3`=Both
`description`	string	Description, max 64 characters; must not contain spaces, `?`, `(`, `;`, or other special characters
`index`	integer	Rule number (read-only, same as the outer `index`)

> `dport` and `to_dport` are string types in the actual response. Up to 64 rules.

2.3.3.2 Save Port Mapping Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_firewall_port_mapping",
  "values": [
    {
      "base": "yruo_firewall_port_mapping",
      "type": "yruo_firewall_port_mapping",
      "index": 1,
      "value": {
        "src": "0.0.0.0/0",
        "dport": "8080",
        "to_dst": "192.168.1.100",
        "to_dport": "80",
        "protocol": 1,
        "description": "web"
      }
    }
  ]
}
```

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code

Field	Type	Description
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.3.4 DMZ

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

2.3.4.1 Query DMZ Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_firewall_dmz",
  "values": [
    { "base": "yruo_firewall_dmz" }
  ]
}
```

Response

```
{
  "id": 11,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
```

```
      "get": [
        {
          "type": "yruo_firewall_dmz",
          "index": 1,
          "value": {
            "src": "",
            "dmz": "",
            "enable": 0
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_firewall_dmz` **type (type=**`yruo_firewall_dmz`**, index=**`1`**)**

Field	Type	Description
`enable`	integer	DMZ enable, `0`=disabled, `1`=enabled
`dmz`	string	DMZ host IP (IPv4), max 15 characters; required when `enable`=`1`
`src`	string	Allowed source IP/mask, max 127 characters, default `0.0.0.0`; required when `enable`=`1`

2.3.4.2 Save DMZ Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_firewall_dmz",
  "values": [
    {
      "base": "yruo_firewall_dmz",
      "type": "yruo_firewall_dmz",
      "index": 1,
      "value": {
        "enable": 1,
        "dmz": "192.168.1.100",
        "src": "0.0.0.0"
      }
    }
  ]
}
```

The `value` object only needs to include the changed fields.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.3.5 MAC Binding

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

---

2.3.5.1 Query MAC Binding Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_firewall_mac_binding",
  "values": [
    { "base": "yruo_firewall_mac_binding" }
  ]
}
```

Response

`result[0].get[]` each item corresponds to one MAC-IP binding rule; `index` is the rule number.


```
{
  "id": 14,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_firewall_mac_binding",
          "index": 1,
          "value": {
            "ip": "2.2.2.2",
            "mac": "00:00:00:00:00:00",
            "description": "33"
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`yruo_firewall_mac_binding` **type (type=**`yruo_firewall_mac_binding`**, index=rule number)**

Field	Type	Description
`mac`	string	MAC address, format `xx:xx:xx:xx:xx:xx`, max 17 characters
`ip`	string	Bound IPv4 address, max 15 characters; unique within the same list
`description`	string	Description, max 64 characters

> Up to 5 rules.

2.3.5.2 Save MAC Binding Configuration

Request


```
{
  "id": 1,
  "function": "set",
  "core": "yruo_firewall_mac_binding",
  "values": [
    {
```

```
      "base": "yruo_firewall_mac_binding",
      "type": "yruo_firewall_mac_binding",
      "index": 1,
      "value": {
        "mac": "AA:BB:CC:DD:EE:FF",
        "ip": "192.168.1.100",
        "description": "my-device"
      }
    }
  ]
}
```

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.3.6 Custom Rules

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

2.3.6.1 Query Custom Rules Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_firewall_custom",
  "values": [
    { "base": "yruo_firewall_custom" }
  ]
}
```

Response

```
{
  "id": 17,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_firewall_custom",
          "index": "qwerty",
          "value": {
            "list": [
              {
                "id": "qwertymnwn1zwh",
                "rule": "Eg: -t filter -I INPUT -s 192.168.3.240 -j DROP",
                "desc": "desc"
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_firewall_custom` **type (type=**`yruo_firewall_custom`**, index=random string)**

Field	Type	Description
`list`	array	Custom iptables rule list, up to 64 entries, see table below

`list` **entry fields**

Field	Type	Description
`id`	string	Rule unique identifier (random string, system-generated)
`rule`	string	iptables rule command, max 255 characters, example: `-t filter -I INPUT -s 192.168.3.240 -j DROP`
`desc`	string	Description, max 63 characters; optional

2.3.6.2 Save Custom Rules Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_firewall_custom",
  "values": [
    {
      "base": "yruo_firewall_custom",
      "type": "yruo_firewall_custom",
      "index": "qwerty",
      "value": {
        "list": [
          {
            "id": "qwertymnwn1zwh",
            "rule": "-t filter -I INPUT -s 192.168.3.240 -j DROP",
            "desc": "block host"
          }
        ]
      }
    }
  ]
}
```

Submit the full `list` array (full replacement).

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.3.7 SPI

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

2.3.7.1 Query SPI Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_firewall_spi",
  "values": [
    { "base": "yruo_firewall_spi" }
  ]
}
```

Response

```
{
  "id": 18,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_firewall_spi",
          "index": 1,
          "value": {
            "filter_proxy": 0,
            "filter_cookies": 0,
            "filter_java": 0,
```

```
            "filter_active_x": 0,
            "filter_ping": 1,
            "filter_snmp": 1,
            "filter_multicast": 1,
            "filter_ident": 0,
            "filter_wan_redirection": 1,
            "filter_ipsec": 0,
            "filter_openvpn": 0,
            "spi_fierwall": 0
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_firewall_spi` **type (type=`yruo_firewall_spi`, index=`1`)**

All fields are integer, `0`=disabled, `1`=enabled.

Field	Description
`spi_fierwall`	SPI firewall master switch (note: the firmware field name is spelled `fierwall`, not `firewall`)
`filter_proxy`	Filter Proxy
`filter_cookies`	Filter Cookies
`filter_java`	Filter Java
`filter_active_x`	Filter ActiveX
`filter_ping`	Filter Ping (WAN port)
`filter_snmp`	Filter SNMP (WAN port)
`filter_multicast`	Filter Multicast
`filter_ident`	Filter Ident
`filter_wan_redirection`	Filter WAN Redirection
`filter_ipsec`	Filter IPsec
`filter_openvpn`	Filter OpenVPN

2.3.7.2 Save SPI Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_firewall_spi",
  "values": [
    {
      "base": "yruo_firewall_spi",
      "type": "yruo_firewall_spi",
      "index": 1,
      "value": {
        "spi_fierwall": 1,
        "filter_ping": 1,
        "filter_snmp": 1,
        "filter_multicast": 1,
        "filter_wan_redirection": 1,
        "filter_proxy": 0,
        "filter_cookies": 0,
        "filter_java": 0,
        "filter_active_x": 0,
        "filter_ident": 0,
        "filter_ipsec": 0,
        "filter_openvpn": 0
      }
    }
  ]
}
```

The `value` object only needs to include the changed fields.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

2.4 Qos

2.4.1 Download Bandwidth Control

2.4.1.1 Get Download Bandwidth Control Configuration

Request

```
POST /cgi
x-session-id: <td>
```

```
{
  "core": "yruo_qos_download",
  "function": "get",
  "values": [{ "base": "yruo_qos_download" }]
}
```

Response

```
{
  "result": [{
    "get": [{
      "type": "yruo_qos_download",
      "index": 1,
      "value": {
        "total_bandwidth": "0",
        "enable": 0,
        "class_list": [],
        "rule_list": []
      }
    }]
  }]
}
```

value field descriptions

Field	Type	Description
`enable`	integer	Feature switch, 0=disabled, 1=enabled
`total_bandwidth`	string	Download total bandwidth, unit kbits/s, range 1–8000000; `"0"` means not configured
`default_class`	string	Default traffic class name, value is one of the `class_name` values in `class_list`
`class_list`	array	Traffic class list, up to 10 entries, see table below
`rule_list`	array	Rule list, up to 20 entries, see table below

> `default_class` is present only when `class_list` is non-empty.

class_list entry fields

Field	Type	Description
`class_name`	string	Class name, max 32 characters, unique within the same list
`percent`	integer	Guaranteed bandwidth percentage (%), range 1–100; the sum of all entries must equal 100
`max`	integer	Maximum bandwidth limit, unit kbits/s, must not exceed `total_bandwidth`; optional
`min`	integer	Minimum guaranteed bandwidth, unit kbits/s, ≥0 and must be less than `max`; optional

rule_list entry fields

Field	Type	Description
`rule_name`	string	Rule name, max 32 characters, unique within the same list
`source`	string	Source IP/mask, format like `192.168.1.0/24`; optional
`sport`	integer	Source port, 0–65535; optional
`dest`	string	Destination IP/mask, format like `192.168.1.0/24`; optional
`dport`	integer	Destination port, 0–65535; optional
`protocol`	integer	Protocol, 0=any, 1=TCP, 2=UDP, 3=ICMP, 4=GRE
`attach_class`	string	Associated class name, must correspond to one of the `class_name` values in `class_list`

2.4.1.2 Set Download Bandwidth Control Configuration

Request

```
POST /cgi
x-session-id: <td>
```

```
{
  "core": "yruo_qos_download",
  "function": "set",
  "values": [{
    "type": "yruo_qos_download",
    "index": 1,
    "value": {
      "enable": 1,
      "total_bandwidth": "10000",
      "default_class": "default",
      "class_list": [
        {
          "class_name": "default",
          "percent": 60,
          "max": 8000,
```

```
          "min": 1000
        },
        {
          "class_name": "high",
          "percent": 40
        }
      ],
      "rule_list": [
        {
          "rule_name": "rule1",
          "source": "192.168.1.0/24",
          "sport": 0,
          "dest": "",
          "dport": 80,
          "protocol": 1,
          "attach_class": "high"
        }
      ]
    }
  }]
}
```

> The SET `values` array structure matches the GET response's `get` array. Submit the full `class_list` and `rule_list` (full replacement).

2.4.2 Upload Bandwidth Control

2.4.2.1 Get Upload Bandwidth Control Configuration

Request

```
POST /cgi
x-session-id: <td>
```

```
{
  "core": "yruo_qos_upload",
  "function": "get",
  "values": [{ "base": "yruo_qos_upload" }]
}
```

Response

```
{
  "result": [{
    "get": [{
      "type": "yruo_qos_upload",
      "index": 1,
      "value": {
        "total_bandwidth": "0",
        "enable": 0,
        "class_list": [],
        "rule_list": []
      }
```

```
    }]
  }]
}
```

value field descriptions

Field	Type	Description
`enable`	integer	Feature switch, 0=disabled, 1=enabled
`total_bandwidth`	string	Upload total bandwidth, unit kbits/s, range 1–8000000; `"0"` means not configured
`default_class`	string	Default traffic class name, value is one of the `class_name` values in `class_list`
`class_list`	array	Traffic class list, up to 10 entries, see table below
`rule_list`	array	Rule list, up to 20 entries, see table below

> `default_class` is present only when `class_list` is non-empty.

class_list entry fields

Field	Type	Description
`class_name`	string	Class name, max 32 characters, unique within the same list
`percent`	integer	Guaranteed bandwidth percentage (%), range 0–100; the sum of all entries must equal 100
`max`	integer	Maximum bandwidth limit, unit kbits/s, must not exceed `total_bandwidth`; optional
`min`	integer	Minimum guaranteed bandwidth, unit kbits/s, ≥0 and must be less than `max`; optional

rule_list entry fields

Field	Type	Description
`rule_name`	string	Rule name, max 32 characters, unique within the same list
`source`	string	Source IP/mask, format like `192.168.1.0/24`; optional
`sport`	integer	Source port, 0–65535; optional
`dest`	string	Destination IP/mask, format like `192.168.1.0/24`; optional
`dport`	integer	Destination port, 0–65535; optional
`protocol`	integer	Protocol, 0=any, 1=TCP, 2=UDP, 3=ICMP, 4=GRE
`attach_class`	string	Associated class name, must correspond to one of the `class_name` values in `class_list`

2.4.2.2 Set Upload Bandwidth Control Configuration

Request

```
POST /cgi
x-session-id: <td>
```

```
{
  "core": "yruo_qos_upload",
  "function": "set",
  "values": [{
    "type": "yruo_qos_upload",
    "index": 1,
    "value": {
      "enable": 1,
      "total_bandwidth": "5000",
      "default_class": "default",
      "class_list": [
        {
          "class_name": "default",
          "percent": 60,
          "max": 4000,
          "min": 500
        },
        {
          "class_name": "high",
          "percent": 40
        }
      ],
      "rule_list": [
        {
          "rule_name": "rule1",
          "source": "192.168.1.0/24",
          "sport": 0,
          "dest": "",
          "dport": 443,
          "protocol": 1,
          "attach_class": "high"
        }
      ]
    }
  }]
}
```

> The SET `values` array structure matches the GET response's `get` array. Submit the full `class_list` and `rule_list` (full replacement).
>
> Upload and download configurations have the same structure; only the `core` / `type` fields differ (`yruo_qos_upload` vs `yruo_qos_download`). The minimum value of `percent` is 0 (download is 1).

2.5 VPN

2.5.1 DMVPN

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.1.1 Get DMVPN Configuration

Request

```
{
  "core": "yruo_vpn_dmvpn",
  "function": "get",
  "values": [{ "base": "yruo_vpn_dmvpn" }]
}
```

Response

```
{
  "id": 19,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_vpn_dmvpn",
          "index": 1,
          "value": {
            "interface_list": ["SIM1-APN1", "WAN/LAN5"],
            "local_subnet": "",
            "local_subnet_mask": "",
            "remote_subnet": "",
            "remote_subnet_mask": "",
            "remote_subnet_list": [],
            "local_subnet_list": [],
```

```
            "lifetime": 3600,
            "pfs_group": 0,
            "mode": 0,
            "protocol": 0,
            "sa_algorithm": 4,
            "version": 1,
            "authentication_type_local": 0,
            "secrets_local": "222",
            "dpd_interval": 30,
            "dpd_timeout": 150,
            "negotiation_mode": 0,
            "local_id_type": 0,
            "remote_id_type": 0,
            "local_id": "",
            "remote_id": "",
            "enable_xauth": 0,
            "enable_comp": 0,
            "encryption_algorithm": 2,
            "authentication_algorithm": 0,
            "dh_group": 0,
            "ike_lifetime": 10800,
            "local_ip_type": 1,
            "local_ip": "10.224.145.147",
            "cust_interface": "SIM1-APN1",
            "gre_ip": "11.2.0.2",
            "gre_mask": "255.255.255.0",
            "gre_mtu": 1400,
            "gre_key": "",
            "hub_ip": "192.168.50.54",
            "hub_gre_ip": "10.0.0.2",
            "cisco_secret": "",
            "nhrp_time": 7200,
            "enable": 1
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`yruo_vpn_dmvpn` **value fields**

Field	Type	Description
`interface_list`	string[]	List of available interfaces (read-only, used to populate the `cust_interface` dropdown options)
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`hub_ip`	string	Hub IP address or domain
`local_ip_type`	number	Local IP acquisition method, see enum table

Field	Type	Description
`local_ip`	string	Local IP (manually entered when `local_ip_type=0`)
`cust_interface`	string	Local interface (selected from `interface_list` when `local_ip_type=1`)
`hub_gre_ip`	string	Hub GRE IP
`gre_ip`	string	Local GRE IP
`gre_mask`	string	GRE netmask
`gre_mtu`	number	GRE MTU, range 64–1500
`gre_key`	string	GRE key (SET only, GET returns empty string)
`negotiation_mode`	number	IKE negotiation mode, see enum table
`encryption_algorithm`	number	IKE encryption algorithm, see enum table
`authentication_algorithm`	number	IKE authentication algorithm, see enum table
`dh_group`	number	DH group, see enum table
`secrets_local`	string	Pre-shared key (PSK)
`local_id_type`	number	Local ID type, see enum table
`local_id`	string	Local ID (effective when `local_id_type≠0`)
`remote_id_type`	number	Peer ID type, see enum table
`remote_id`	string	Peer ID
`ike_lifetime`	number	IKE lifetime (seconds), range 60–86400
`sa_algorithm`	number	SA algorithm, see enum table
`pfs_group`	number	PFS group, see enum table
`lifetime`	number	SA lifetime (seconds), range 60–86400
`dpd_interval`	number	DPD detection interval (seconds), range 1–60
`dpd_timeout`	number	DPD timeout (seconds), range 10–3600
`cisco_secret`	string	Cisco NHRP key (SET only, GET returns empty string)
`nhrp_time`	number	NHRP registration time (seconds), range 60–86400
`mode`	number	Mode (reserved field)
`protocol`	number	Protocol (reserved field)

Field	Type	Description
`version`	number	IKE version (reserved field)
`authentication_type_local`	number	Local authentication type (reserved field)
`enable_xauth`	number	Whether XAuth is enabled (reserved field)
`enable_comp`	number	Whether compression is enabled (reserved field)
`local_subnet`	string	Local subnet (reserved field)
`local_subnet_mask`	string	Local subnet mask (reserved field)
`remote_subnet`	string	Peer subnet (reserved field)
`remote_subnet_mask`	string	Peer subnet mask (reserved field)
`remote_subnet_list`	array	Peer subnet list (reserved field)
`local_subnet_list`	array	Local subnet list (reserved field)

Enum values

`local_ip_type`

Value	Meaning
`0`	Manual
`1`	Auto (from interface)

`negotiation_mode`

Value	Meaning
`0`	Main mode
`1`	Aggressive mode

`encryption_algorithm`

Value	Meaning
`0`	DES (insecure)
`1`	3DES (insecure)
`2`	AES-128
`3`	AES-192
`4`	AES-256

`authentication_algorithm`

Value	Meaning
`0`	MD5
`1`	SHA1
`2`	SHA2-256

`dh_group`

Value	Meaning
`0`	DH1
`1`	DH2
`2`	DH5
`3`	DH14
`4`	DH15

`local_id_type` / `remote_id_type`

Value	Meaning
`0`	IP
`1`	FQDN
`2`	User FQDN
`3`	ASN1DN

`sa_algorithm` (known enum values, may be incomplete)

Value	Meaning
`0`	DES-MD5
`1`	DES-SHA1
`2`	3DES-MD5
`3`	3DES-SHA1
`4`	AES128-MD5
`5`	AES128-SHA1
`6`	AES192-MD5
`7`	AES192-SHA1
`8`	AES256-MD5

Value	Meaning
`9`	AES256-SHA1
`12`	AES128-SHA256
`13`	AES192-SHA256
`14`	AES256-SHA256

`pfs_group`

Value	Meaning
`0`	Disabled
`1`	DH1
`2`	DH2
`3`	DH5

2.5.1.2 Save DMVPN Configuration

Request

```
{
  "core": "yruo_vpn_dmvpn",
  "function": "set",
  "values": [
    {
      "base": "yruo_vpn_dmvpn",
      "type": "yruo_vpn_dmvpn",
      "index": 1,
      "value": {
        "enable": 1,
        "hub_ip": "192.168.50.54"
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.2 IPsec Server

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.2.1 Get IPsec Server Configuration

Request

```
{
  "core": "yruo_vpn_ipsec_server",
  "function": "get",
  "values": [{ "base": "yruo_vpn_ipsec_server" }]
}
```

Response

```
{
  "id": 20,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_vpn_ipsec_server",
          "index": 1,
          "value": {
            "local_subnet": "",
            "local_subnet_mask": "",
            "remote_subnet": "",
            "remote_subnet_mask": "",
            "remote_subnet_list": [],
            "local_subnet_list": [],
            "lifetime": 3600,
            "pfs_group": 0,
            "mode": 0,
```

```
            "protocol": 0,
            "sa_encrypt_algo": "esp-aes128",
            "sa_auth_algo": "md5",
            "version": 1,
            "authentication_type_local": 0,
            "dpd_interval": 30,
            "dpd_timeout": 150,
            "negotiation_mode": 0,
            "local_id_type": 0,
            "remote_id_type": 0,
            "local_id": "",
            "remote_id": "",
            "enable_xauth": 0,
            "enable_comp": 0,
            "encryption_algorithm": 2,
            "authentication_algorithm": 0,
            "dh_group": 0,
            "ike_lifetime": 10800,
            "ipsec_over_vpn": 0,
            "enable": 0,
            "margin_time": 100,
            "psk_list": [],
            "xauth_list": []
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`yruo_vpn_ipsec_server` **value fields**

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`mode`	number	Encapsulation mode, `0`=tunnel, `1`=transport
`protocol`	number	Protocol, `0`=ESP, `1`=AH
`local_subnet_list`	string[]	Local subnet list (CIDR format, valid in tunnel mode)
`remote_subnet_list`	string[]	Peer subnet list (CIDR format, valid in tunnel mode)
`local_id_type`	number	Local ID type, see enum table
`local_id`	string	Local ID (effective when `local_id_type≠0`)
`remote_id_type`	number	Peer ID type, see enum table
`remote_id`	string	Peer ID

Field	Type	Description
`version`	number	IKE version, `1`=IKEv1, `2`=IKEv2
`negotiation_mode`	number	Negotiation mode, `0`=main mode, `1`=aggressive mode (valid for IKEv1)
`encryption_algorithm`	number	IKE encryption algorithm, see enum table
`authentication_algorithm`	number	IKE authentication algorithm, see enum table
`dh_group`	number	DH group, see enum table
`authentication_type_local`	number	Local authentication type, `0`=PSK, `1`=CA
`enable_xauth`	number	Whether XAuth is enabled (valid for IKEv1), `0` / `1`
`xauth_list`	object[]	XAuth user list, see sub-field descriptions
`psk_list`	object[]	PSK list, see sub-field descriptions
`ike_lifetime`	number	IKE lifetime (seconds), range 60–86400
`sa_encrypt_algo`	string	SA encryption algorithm, see enum table
`sa_auth_algo`	string	SA authentication algorithm, see enum table
`pfs_group`	number	PFS group, see enum table
`lifetime`	number	SA lifetime (seconds), range 60–86400
`dpd_interval`	number	DPD detection interval (seconds)
`dpd_timeout`	number	DPD timeout (seconds)
`enable_comp`	number	Whether compression is enabled, `0` / `1`
`margin_time`	number	SA early renewal time (seconds)
`ipsec_over_vpn`	number	IPsec over VPN, `0`=disabled, `1`=L2TP, `2`=GRE
`local_subnet`	string	Local subnet (legacy field, replaced by `local_subnet_list`)
`local_subnet_mask`	string	Local subnet mask (legacy field)
`remote_subnet`	string	Peer subnet (legacy field)
`remote_subnet_mask`	string	Peer subnet mask (legacy field)

`xauth_list` sub-fields

Field	Type	Description
`username`	string	Username

Field	Type	Description
`password`	string	Password (SET only)
`old_username`	string	Original username (passed on modification, used to identify the record)
`old_password`	string	Original password (passed on modification)

`psk_list` sub-fields

Field	Type	Description
`selector`	string	Selector (peer identifier)
`psk`	string	Pre-shared key
`old_selector`	string	Original selector (passed on modification)
`old_psk`	string	Original PSK (passed on modification)

Enum values

`local_id_type` / `remote_id_type`

Value	Meaning
`0`	Default (IP)
`1`	ID
`2`	FQDN
`3`	User FQDN

`encryption_algorithm`

Value	Meaning
`0`	DES (insecure)
`1`	3DES (insecure)
`2`	AES-128
`3`	AES-192
`4`	AES-256

`authentication_algorithm`

Value	Meaning
`0`	MD5

Value	Meaning
`1`	SHA1
`2`	SHA2-256

`dh_group`

Value	Meaning
`0`	DH1
`1`	DH2
`2`	DH5
`3`	DH14
`4`	DH15

`sa_encrypt_algo`

Value	Meaning
`"esp-des"`	DES (insecure)
`"esp-3des"`	3DES (insecure)
`"esp-aes128"`	AES-128
`"esp-aes192"`	AES-192
`"esp-aes256"`	AES-256

`sa_auth_algo`

Value	Meaning
`"md5"`	MD5
`"sha1"`	SHA1
`"sha256"`	SHA2-256

`pfs_group`

Value	Meaning
`0`	Disabled
`1`	DH1
`2`	DH2
`3`	DH5

2.5.2.2 Save IPsec Server Configuration

Request

```
{
  "core": "yruo_vpn_ipsec_server",
  "function": "set",
  "values": [
    {
      "base": "yruo_vpn_ipsec_server",
      "type": "yruo_vpn_ipsec_server",
      "index": 1,
      "value": {
        "enable": 1,
        "mode": 0
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.3 IPsec

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.3.1 Get IPsec Client Configuration

Request

```
{
  "core": "yruo_vpn_ipsec_client",
  "function": "get",
  "values": [{ "base": "yruo_vpn_ipsec_client" }]
}
```

Response


```
{
  "id": 21,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_vpn_ipsec_client",
          "index": 1,
          "value": {
            "local_subnet": "",
            "local_subnet_mask": "",
            "remote_subnet": "",
            "remote_subnet_mask": "",
            "remote_subnet_list": [],
            "local_subnet_list": [],
            "lifetime": 3600,
            "pfs_group": 0,
            "mode": 0,
            "protocol": 0,
            "sa_encrypt_algo": "esp-aes128",
            "sa_auth_algo": "md5",
            "version": 1,
            "authentication_type_local": 0,
            "dpd_interval": 30,
            "dpd_timeout": 150,
            "negotiation_mode": 0,
            "local_id_type": 0,
            "remote_id_type": 0,
            "local_id": "",
            "remote_id": "",
            "enable_xauth": 0,
            "enable_comp": 0,
            "encryption_algorithm": 2,
            "authentication_algorithm": 0,
            "dh_group": 0,
            "ike_lifetime": 10800,
            "ipsec_over_vpn": 0,
            "enable": 0,
            "margin_time": 100
          }
```

```
        }
        // index 2, 3 same structure, omitted
      ]
    }
  ]
}
```

Supports up to 3 records (index 1–3).

Field Descriptions

`yruo_vpn_ipsec_client` **value fields**

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`remote_ip`	string	Peer IP or domain (required on SET)
`mode`	number	Encapsulation mode, `0`=tunnel, `1`=transport
`protocol`	number	Protocol, `0`=ESP, `1`=AH
`local_subnet_list`	string[]	Local subnet list (CIDR format, valid in tunnel mode)
`remote_subnet_list`	string[]	Peer subnet list (CIDR format, valid in tunnel mode)
`local_id_type`	number	Local ID type, see enum table
`local_id`	string	Local ID
`remote_id_type`	number	Peer ID type, see enum table
`remote_id`	string	Peer ID
`version`	number	IKE version, `1`=IKEv1, `2`=IKEv2
`negotiation_mode`	number	Negotiation mode, `0`=main mode, `1`=aggressive mode
`encryption_algorithm`	number	IKE encryption algorithm, see enum table
`authentication_algorithm`	number	IKE authentication algorithm, see enum table
`dh_group`	number	DH group, see enum table
`authentication_type_local`	number	Local authentication type, `0`=PSK, `1`=CA
`enable_xauth`	number	Whether XAuth is enabled, `0` / `1`
`ike_lifetime`	number	IKE lifetime (seconds), range 60–86400
`sa_encrypt_algo`	string	SA encryption algorithm, see enum table
`sa_auth_algo`	string	SA authentication algorithm, see enum table

Field	Type	Description
`pfs_group`	number	PFS group, see enum table
`lifetime`	number	SA lifetime (seconds), range 60–86400
`dpd_interval`	number	DPD detection interval (seconds)
`dpd_timeout`	number	DPD timeout (seconds)
`enable_comp`	number	Whether compression is enabled, `0` / `1`
`margin_time`	number	SA early renewal time (seconds)
`ipsec_over_vpn`	number	IPsec over VPN, `0`=disabled, `1`=L2TP, `2`=GRE
`local_subnet`	string	Local subnet (legacy field)
`local_subnet_mask`	string	Local subnet mask (legacy field)
`remote_subnet`	string	Peer subnet (legacy field)
`remote_subnet_mask`	string	Peer subnet mask (legacy field)

> Compared to the IPsec server, the client has no `psk_list` and `xauth_list` fields, but adds `remote_ip`. Enum values are the same as in 2.5.2 IPsec Server.

2.5.3.2 Save IPsec Client Configuration

Request

```
{
  "core": "yruo_vpn_ipsec_client",
  "function": "set",
  "values": [
    {
      "base": "yruo_vpn_ipsec_client",
      "type": "yruo_vpn_ipsec_client",
      "index": 1,
      "value": {
        "enable": 1,
        "remote_ip": "203.0.113.1"
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code

Field	Type	Description
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.4 GRE

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.4.1 Get GRE Configuration

Request

```
{
  "core": "yruo_vpn_gre",
  "function": "get",
  "values": [{ "base": "yruo_vpn_gre" }]
}
```

Response

```
{
  "id": 22,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_vpn_gre",
          "index": 1,
          "value": {
            "local_ip": "",
```

```
            "remote_ip": "",
            "local_virtual_ip": "",
            "local_virtual_mask": "255.255.255.0",
            "remote_virtual_ip": "",
            "remote_subnet": "",
            "remote_mask": "",
            "enable_nat": 1,
            "default_route": 0,
            "mtu": 1500,
            "key": "",
            "enable": 0
          }
        }
        // index 2, 3 same structure, omitted
      ]
    }
  ]
}
```

Supports up to 3 records (index 1–3).

Field Descriptions

`yruo_vpn_gre` **value fields**

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`remote_ip`	string	Peer IP or domain
`local_ip`	string	Local IP
`local_virtual_ip`	string	Local virtual IP
`local_virtual_mask`	string	Local virtual netmask
`remote_virtual_ip`	string	Peer virtual IP
`default_route`	number	Whether to use the default route, `0`=no, `1`=yes
`remote_subnet`	string	Peer subnet (effective when `default_route=0`)
`remote_mask`	string	Peer subnet mask
`mtu`	number	MTU, range 64–1500
`key`	string	GRE key (SET only, GET returns empty string)
`enable_nat`	number	Whether NAT is enabled, `0`=disabled, `1`=enabled

2.5.4.2 Save GRE Configuration

Request

```
{
  "core": "yruo_vpn_gre",
  "function": "set",
  "values": [
    {
      "base": "yruo_vpn_gre",
      "type": "yruo_vpn_gre",
      "index": 1,
      "value": {
        "enable": 1,
        "remote_ip": "203.0.113.1"
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.5 L2TP

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.5.1 Get L2TP Configuration

Request

```
{
  "core": "yruo_vpn_l2tp",
  "function": "get",
  "values": [{ "base": "yruo_vpn_l2tp" }]
}
```

Response

```
{
  "id": 23,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_vpn_l2tp",
          "index": 1,
          "value": {
            "remote_ip": "",
            "user": "",
            "password": "",
            "auth_method": 0,
            "enable_nat": 1,
            "default_route": 0,
            "enable_mppe": 0,
            "ac_comp": 0,
            "p_comp": 0,
            "asyncmap": "ffffffff",
            "mtu": 1500,
            "mru": 1500,
            "echo_interval": 60,
            "max_retries": 0,
            "local_ip": "",
            "peer_ip": "",
            "expert_opts": "",
            "remote_subnet": "",
            "remote_mask": "",
            "key": "",
            "hostname": "",
            "enable": 0
          }
        }
        // index 2, 3 same structure, omitted
      ]
    }
  ]
}
```

Supports up to 3 records (index 1–3).

Field Descriptions

`yruo_vpn_l2tp` **value fields**

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`remote_ip`	string	Server IP or domain
`hostname`	string	Hostname
`user`	string	Username
`password`	string	Password (SET only, GET returns empty string)
`auth_method`	number	Authentication method, see enum table
`default_route`	number	Whether to use the default route, `0`=no, `1`=yes
`remote_subnet`	string	Peer subnet (effective when `default_route=0`)
`remote_mask`	string	Peer subnet mask
`key`	string	L2TP key (SET only, GET returns empty string)
`local_ip`	string	Local IP (advanced option)
`peer_ip`	string	Peer IP (advanced option)
`enable_nat`	number	Whether NAT is enabled, `0`=disabled, `1`=enabled (advanced option)
`enable_mppe`	number	Whether MPPE encryption is enabled, `0`/`1` (advanced option)
`ac_comp`	number	Whether AC compression is enabled, `0`/`1` (advanced option)
`p_comp`	number	Whether protocol compression is enabled, `0`/`1` (advanced option)
`asyncmap`	string	Asynchronous character map, hex string (advanced option)
`mtu`	number	MTU, range 128–1500 (advanced option)
`mru`	number	MRU, range 64–1500 (advanced option)
`echo_interval`	number	Heartbeat interval (seconds), range 0–600 (advanced option)
`max_retries`	number	Maximum retry count, range 0–10 (advanced option)
`expert_opts`	string	Expert options (advanced option)

Enum values

`auth_method` (known enum values, may be incomplete)

Value	Meaning
`0`	Auto
`1`	PAP
`2`	CHAP
`3`	MS-CHAP
`4`	MS-CHAPv2

2.5.5.2 Save L2TP Configuration

Request

```
{
  "core": "yruo_vpn_l2tp",
  "function": "set",
  "values": [
    {
      "base": "yruo_vpn_l2tp",
      "type": "yruo_vpn_l2tp",
      "index": 1,
      "value": {
        "enable": 1,
        "remote_ip": "203.0.113.1",
        "user": "username",
        "password": "secret"
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.6 PPTP

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.6.1 Get PPTP Configuration

Request

```
{
  "core": "yruo_vpn_pptp",
  "function": "get",
  "values": [{ "base": "yruo_vpn_pptp" }]
}
```

Response

```
{
  "id": 24,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_vpn_pptp",
          "index": 1,
          "value": {
            "remote_ip": "",
            "user": "",
            "password": "",
            "auth_method": 0,
            "enable_nat": 1,
            "default_route": 0,
            "enable_mppe": 0,
            "ac_comp": 0,
            "p_comp": 0,
            "asyncmap": "ffffffff",
            "mtu": 1500,
            "mru": 1500,
            "echo_interval": 60,
            "max_retries": 0,
            "local_ip": "",
```

```
            "peer_ip": "",
            "expert_opts": "",
            "remote_subnet": "",
            "remote_mask": "",
            "enable": 0
          }
        }
        // index 2, 3 same structure, omitted
      ]
    }
  ]
}
```

Supports up to 3 records (index 1–3).

Field Descriptions

`yruo_vpn_pptp` value fields

Basically the same as the L2TP fields, except there are no `key` and `hostname` fields.

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`remote_ip`	string	Server IP or domain
`user`	string	Username
`password`	string	Password (SET only, GET returns empty string)
`auth_method`	number	Authentication method, see enum table
`default_route`	number	Whether to use the default route, `0`=no, `1`=yes
`remote_subnet`	string	Peer subnet (effective when `default_route=0`)
`remote_mask`	string	Peer subnet mask
`local_ip`	string	Local IP (advanced option)
`peer_ip`	string	Peer IP (advanced option)
`enable_nat`	number	Whether NAT is enabled, `0`=disabled, `1`=enabled (advanced option)
`enable_mppe`	number	Whether MPPE encryption is enabled, `0`/`1` (advanced option)
`ac_comp`	number	Whether AC compression is enabled, `0`/`1` (advanced option)
`p_comp`	number	Whether protocol compression is enabled, `0`/`1` (advanced option)
`asyncmap`	string	Asynchronous character map, hex string (advanced option)

Field	Type	Description
`mtu`	number	MTU, range 128–1500 (advanced option)
`mru`	number	MRU, range 64–1500 (advanced option)
`echo_interval`	number	Heartbeat interval (seconds), range 0–600 (advanced option)
`max_retries`	number	Maximum retry count, range 0–10 (advanced option)
`expert_opts`	string	Expert options (advanced option)

Enum values

`auth_method` (known enum values, may be incomplete)

Value	Meaning
`0`	Auto
`1`	PAP
`2`	CHAP
`3`	MS-CHAP
`4`	MS-CHAPv2

2.5.6.2 Save PPTP Configuration

Request

```
{
  "core": "yruo_vpn_pptp",
  "function": "set",
  "values": [
    {
      "base": "yruo_vpn_pptp",
      "type": "yruo_vpn_pptp",
      "index": 1,
      "value": {
        "enable": 1,
        "remote_ip": "203.0.113.1",
        "user": "username",
        "password": "secret"
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.7 OpenVPN Client

> The `/cgi` interface uses `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with `multipart/form-data` format.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.7.1 Get OpenVPN Client Configuration

Request

```
{
  "core": "yruo_vpn_openvpn_client",
  "function": "get",
  "values": [{ "base": "yruo_vpn_openvpn_client" }]
}
```

Response

```
{
  "id": 25,
  "model": "UR35",
  "status": 0,
  "result": [
```

```
    {
      "get": [
        {
          "type": "yruo_vpn_openvpn_client",
          "index": 1,
          "value": {
            "remote_ip": "",
            "protocol": 0,
            "config_mode": 0,
            "port": 1194,
            "interface_type": 0,
            "authentication": 0,
            "cipher": 0,
            "local_virtual_ip": "",
            "local_virtual_mask": "",
            "remote_virtual_ip": "",
            "username": "",
            "password": "",
            "tls_authentication": 0,
            "enable_nat": 1,
            "default_route": 0,
            "comp": 1,
            "ping_interval": 60,
            "ping_restart": 300,
            "mtu": 1500,
            "fragment": 1500,
            "log_level": 0,
            "expert_options": "",
            "enable": 0,
            "auth_mode": 0,
            "remote_subnet": []
          }
        }
        // index 2, 3 same structure, omitted
      ]
    }
  ]
}
```

Supports up to 3 records (index 1–3).

Field Descriptions

`yruo_vpn_openvpn_client` value fields

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`config_mode`	number	Configuration mode, `0`=standard, `1`=custom config file
`protocol`	number	Protocol, `0`=UDP, `1`=TCP Client
`remote_ip`	string	Server IP or domain (standard mode)

Field	Type	Description
`port`	number	Port, range 1–65535 (standard mode)
`interface_type`	number	Interface type, `0`=TUN, `1`=TAP
`authentication`	number	Authentication method, see enum table
`local_virtual_ip`	string	Local virtual IP
`local_virtual_mask`	string	Local virtual netmask (TAP mode)
`remote_virtual_ip`	string	Peer virtual IP (TUN mode)
`username`	string	Username (when authentication is username/password)
`password`	string	Password (SET only, GET returns empty string)
`tls_authentication`	number	TLS authentication, `0`=none, `1`=TLS Auth, `2`=TLS Crypt
`default_route`	number	Whether to use the default route, `0`=no, `1`=yes
`enable_nat`	number	Whether NAT is enabled, `0`=disabled, `1`=enabled
`comp`	number	Compression, `0`=none, `1`=LZO
`ping_interval`	number	Ping interval (seconds), range 10–1800
`ping_restart`	number	Ping restart timeout (seconds), range 60–3600
`cipher`	number	Encryption algorithm, see enum table
`auth_mode`	number	HMAC authentication algorithm, see enum table
`mtu`	number	MTU, range 128–1500
`fragment`	number	Fragment size, range 128–1500 (UDP mode)
`log_level`	number	Log level, see enum table
`expert_options`	string	Expert options
`remote_subnet`	object[]	Remote route list, see sub-field descriptions

`remote_subnet` sub-fields

Field	Type	Description
`remote_subnet`	string	Remote subnet IP
`remote_mask`	string	Remote subnet mask
`old_remote_subnet`	string	Original subnet IP (passed on modification)
`old_remote_mask`	string	Original subnet mask (passed on modification)

Enum values

`authentication`

Value	Meaning
`0`	None
`1`	Pre-shared key
`2`	Username/password
`3`	Certificate
`4`	Username + certificate

`cipher`

Value	Meaning
`0`	None
`1`	BF-CBC (insecure)
`2`	DES-CBC (insecure)
`3`	DES-EDE3 (insecure)
`4`	AES-128-CBC
`5`	AES-192-CBC
`6`	AES-256-CBC

`auth_mode`

Value	Meaning
`0`	None
`1`	MD5
`2`	SHA1
`3`	SHA256
`4`	SHA512

`log_level`

Value	Meaning
`0`	Error
`1`	Warning

Value	Meaning
`2`	Notice
`3`	Debug

2.5.7.2 Save OpenVPN Client Configuration

Request

```
{
  "core": "yruo_vpn_openvpn_client",
  "function": "set",
  "values": [
    {
      "base": "yruo_vpn_openvpn_client",
      "type": "yruo_vpn_openvpn_client",
      "index": 1,
      "value": {
        "enable": 1,
        "remote_ip": "203.0.113.1",
        "port": 1194
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.7.3 Upload Custom Config File

When `config_mode=1`, you must first upload a `.conf` or `.ovpn` config file.

Request

`POST https://<device-ip>/cgi-bin/file-import`, `Content-Type: multipart/form-data`

Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	`type=openvpn_client&file=custom config`

Field	Description
`size`	File size (bytes)
`file`	File binary content (`.conf` or `.ovpn`)

Success response

```
{ "size": 1234, "checksum": "abc123", "filename": "openvpn_1-custom.conf" }
```

Failure response

```
{ "templet_code": 10000001, "params_count": 0, "templet_params": [] }
```

2.5.8 OpenVPN Server

> The `/cgi` interface uses `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with `multipart/form-data` format.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.8.1 Get OpenVPN Server Configuration

Request

```
{
  "core": "yruo_vpn_openvpn_server",
  "function": "get",
  "values": [{ "base": "yruo_vpn_openvpn_server" }]
}
```

Response


```
{
  "id": 26,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_vpn_openvpn_server",
          "index": 1,
          "value": {
            "enable": 0,
            "config_mode": 0,
            "protocol": 0,
            "port": 1194,
            "listen_ip": "",
            "interface_type": 0,
            "authentication": 0,
            "local_virtual_ip": "",
            "local_virtual_mask": "",
            "remote_virtual_ip": "",
            "client_virtual_subnet": "",
            "client_virtual_mask": "",
            "renegotiation_time": 0,
            "max_clients": 10,
            "default_route": 0,
            "tls_authentication": 0,
            "crl": 0,
            "client_to_client": 0,
            "client_dup": 0,
            "enable_nat": 1,
            "comp": 1,
            "ping_interval": 60,
            "ping_restart": 300,
            "cipher": 0,
            "auth_mode": 0,
            "mtu": 1500,
            "fragment": 1500,
            "log_level": 0,
            "expert_options": "",
            "account": [],
            "local_subnet": [],
            "client_subnet": []
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_vpn_openvpn_server` **value fields**

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`config_mode`	number	Configuration mode, `0`=standard, `1`=custom config file
`protocol`	number	Protocol, `0`=UDP, `1`=TCP Client
`port`	number	Port, range 1–65535
`listen_ip`	string	Listen IP (standard mode)
`interface_type`	number	Interface type, `0`=TUN, `1`=TAP
`authentication`	number	Authentication method, see enum table
`local_virtual_ip`	string	Local virtual IP
`local_virtual_mask`	string	Local virtual netmask (TAP mode)
`remote_virtual_ip`	string	Peer virtual IP (TUN mode)
`client_virtual_subnet`	string	Client virtual subnet (when authentication is username/password or certificate)
`client_virtual_mask`	string	Client virtual subnet mask
`renegotiation_time`	number	Renegotiation time (seconds), range 0–86400
`max_clients`	number	Maximum clients, range 1–128
`default_route`	number	Whether to use the default route, `0`=no, `1`=yes
`tls_authentication`	number	TLS authentication, `0`=none, `1`=TLS Auth, `2`=TLS Crypt
`crl`	number	Whether CRL is enabled, `0`=disabled, `1`=enabled
`client_to_client`	number	Whether client-to-client is allowed, `0`=disabled, `1`=enabled
`client_dup`	number	Whether duplicate clients are allowed, `0`=disabled, `1`=enabled
`enable_nat`	number	Whether NAT is enabled, `0`=disabled, `1`=enabled
`comp`	number	Compression, `0`=none, `1`=LZO
`ping_interval`	number	Ping interval (seconds), range 10–1800
`ping_restart`	number	Ping restart timeout (seconds), range 60–3600
`cipher`	number	Encryption algorithm, see enum table

Field	Type	Description
`auth_mode`	number	HMAC authentication algorithm, see enum table
`mtu`	number	MTU, range 64–1500
`fragment`	number	Fragment size, range 64–1500 (UDP mode)
`log_level`	number	Log level, see enum table
`expert_options`	string	Expert options
`account`	object[]	User account list, see sub-field descriptions
`local_subnet`	object[]	Local route list, see sub-field descriptions
`client_subnet`	object[]	Client subnet list, see sub-field descriptions

`account` sub-fields

Field	Type	Description
`username`	string	Username
`password`	string	Password
`old_username`	string	Original username (passed on modification)
`old_password`	string	Original password (passed on modification)

`local_subnet` sub-fields

Field	Type	Description
`local_ip`	string	Local subnet IP
`local_mask`	string	Local subnet mask
`old_local_ip`	string	Original subnet IP (passed on modification)
`old_local_mask`	string	Original subnet mask (passed on modification)

`client_subnet` sub-fields

Field	Type	Description
`client_name`	string	Client name (certificate CN)
`client_ip`	string	Client subnet IP
`client_mask`	string	Client subnet mask
`old_client_name`	string	Original client name (passed on modification)
`old_client_ip`	string	Original subnet IP (passed on modification)

Field	Type	Description
`old_client_mask`	string	Original subnet mask (passed on modification)

Enum values

`authentication`

Value	Meaning
`0`	None
`1`	Pre-shared key
`2`	Username/password
`3`	Certificate
`4`	Username + certificate

`cipher`

Value	Meaning
`0`	None
`1`	BF-CBC (insecure)
`2`	DES-CBC (insecure)
`3`	DES-EDE3 (insecure)
`4`	AES-128-CBC
`5`	AES-192-CBC
`6`	AES-256-CBC

`auth_mode`

Value	Meaning
`0`	None
`1`	MD5
`2`	SHA1
`3`	SHA256
`4`	SHA512

`log_level`

Value	Meaning
`0`	Error
`1`	Warning
`2`	Notice
`3`	Debug

2.5.8.2 Save OpenVPN Server Configuration

Request

```
{
  "core": "yruo_vpn_openvpn_server",
  "function": "set",
  "values": [
    {
      "base": "yruo_vpn_openvpn_server",
      "type": "yruo_vpn_openvpn_server",
      "index": 1,
      "value": {
        "enable": 1,
        "port": 1194
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.8.3 Upload Custom Config File

When `config_mode=1`, you must first upload a `.conf` or `.ovpn` config file.

Request

`POST https://<device-ip>/cgi-bin/file-import`, `Content-Type: multipart/form-data`

Field	Description
`sessionid`	Session credential, same as the `x-session-id` value

Field	Description
`filename`	`type=openvpn_server&file=custom config`
`size`	File size (bytes)
`file`	File binary content (`.conf` or `.ovpn`)

Success response

```
{ "size": 1234, "checksum": "abc123", "filename": "openvpn_server-custom.conf" }
```

Failure response

```
{ "templet_code": 10000001, "params_count": 0, "templet_params": [] }
```

2.5.9 WireGuard

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.9.1 Get WireGuard Configuration

Request

```
{
  "core": "yruo_wireguard",
  "function": "get",
  "values": [{ "base": "yruo_wireguard" }]
}
```

Response


```
{
  "id": 27,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_wireguard",
          "index": 1,
          "value": {
            "enable": 0,
            "name": "wg0",
            "custom_private_key": 0,
            "private_key": "",
            "public_key": "",
            "address": "",
            "port": 51820,
            "dns": "",
            "mtu": 1420,
            "peers": []
          }
        }
        // index 2, 3 same structure, omitted
      ]
    }
  ]
}
```


Supports up to 3 records (index 1–3).

Field Descriptions

`yruo_wireguard` **value fields**

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`name`	string	Interface name (read-only, e.g. `wg0`, `wg1`, `wg2`)
`custom_private_key`	number	Whether to use a custom private key, `0`=no, `1`=yes
`private_key`	string	Private key (SET only, GET returns empty string; effective when `custom_private_key=1`, fixed length 44 characters)
`public_key`	string	Public key (read-only, generated by the device)
`address`	string	Local address (CIDR format, e.g. `10.0.0.1/24`)
`port`	number	Listen port, range 1–65535
`dns`	string	DNS server, multiple separated by commas

Field	Type	Description
`mtu`	number	MTU, range 68–1500
`peers`	object[]	Peer list, see sub-field descriptions

`peers` sub-fields

Field	Type	Description
`id`	string	Peer unique identifier (read-only, generated by the device)
`name`	string	Peer name
`public_key`	string	Peer public key (fixed length 44 characters)
`allowed_ips`	string[]	Allowed IP list (CIDR format, up to 8 entries)
`route_enable`	number	Whether to add a route, `0`=no, `1`=yes
`psk`	string	Pre-shared key (SET only, GET returns empty string; fixed length 44 characters, optional)
`endpoint`	string	Peer IP or domain (filled together with `port`)
`port`	number	Peer port, range 1–65535 (filled together with `endpoint`)
`keepalive`	number	Keepalive interval (seconds), range 0–65535
`enable`	number	Whether to enable this peer, `0`=disabled, `1`=enabled

---

2.5.9.2 Save WireGuard Base Configuration

Request

```
{
  "core": "yruo_wireguard",
  "function": "set",
  "values": [
    {
      "base": "yruo_wireguard",
      "type": "yruo_wireguard",
      "index": 1,
      "value": {
        "enable": 1,
        "address": "10.0.0.1/24",
        "port": 51820
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.5.9.3 Add Peer

Request

```
{
  "core": "yruo_wireguard",
  "function": "set",
  "values": [
    {
      "base": "yruo_wireguard",
      "index": 1,
      "value": {
        "peers": [
          {
            "action": 1,
            "name": "peer1",
            "public_key": "YNXzDO7JlEtDFZ+UGrYLOeBKvJpLwiMO9HWs7TQeyF8=",
            "allowed_ips": ["10.0.0.2/32"],
            "route_enable": 1,
            "endpoint": "203.0.113.1",
            "port": 51820,
            "keepalive": 25,
            "enable": 1
          }
        ]
      }
    }
  ]
}
```

`action` field description:

Value	Meaning
`1`	Add peer
`2`	Modify peer
`3`	Delete peer

2.5.9.4 Modify Peer

Request

```
{
  "core": "yruo_wireguard",
  "function": "set",
  "values": [
    {
      "base": "yruo_wireguard",
      "index": 1,
      "value": {
        "peers": [
          {
            "action": 2,
            "name": "peer1-renamed",
            "old_name": "peer1",
            "public_key": "YNXzDO7JlEtDFZ+UGrYLOeBKvJpLwiMO9HWs7TQeyF8=",
            "allowed_ips": ["10.0.0.2/32"],
            "route_enable": 1,
            "keepalive": 25,
            "enable": 1
          }
        ]
      }
    }
  ]
}
```

> `old_name` is passed only when the name changes; the value is the peer name before modification.

2.5.9.5 Delete Peer

Request

```
{
  "core": "yruo_wireguard",
  "function": "set",
  "values": [
    {
      "base": "yruo_wireguard",
      "index": 1,
      "value": {
        "peers": [{ "name": "peer1", "action": 3 }]
      }
    }
  ]
}
```

2.5.10 ZeroTier

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.10.1 Get ZeroTier Configuration

Request

```
{
  "core": "yruo_zerotier",
  "function": "get",
  "values": [{ "base": "yruo_zerotier" }]
}
```

Response

```
{
  "id": 28,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "node_id": "a1b2c3d4e5",
      "node_id_status": 3,
      "get": [
        {
          "type": "yruo_zerotier",
          "index": 1,
          "value": {
            "enable": 1,
            "name": "Network1",
            "status": 1,
            "node_id": "a1b2c3d4e5",
            "network_id": "a1b2c3d4e5f6a7b8",
            "interface": "ztmosfxg001",
            "allow_managed": 1,
```

```
            "ipaddr": "192.168.100.1/24",
            "allow_global": 0,
            "allow_default": 0
          }
        }
      ]
    }
  ]
}
```

Supports up to 3 network records.

Top-level special fields (`result[0]` level)

Field	Type	Description
`node_id`	string	Device ZeroTier Node ID (10-character hex string)
`node_id_status`	number	Node ID status, see enum table

`node_id_status`

Value	Meaning
`1`	Generating
`2`	Generation failed
`3`	Generation successful

`yruo_zerotier` value field descriptions

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`name`	string	Network name
`status`	number	Connection status, see enum table (read-only)
`node_id`	string	Device Node ID (read-only)
`network_id`	string	ZeroTier network ID (16-character hex string)
`interface`	string	Virtual network interface name (read-only)
`allow_managed`	number	Whether managed addresses are allowed, `0`=no, `1`=yes
`ipaddr`	string	Manually specified IP address (CIDR format, effective when `allow_managed=0`)
`allow_global`	number	Whether global IP assignment is allowed, `0`=no, `1`=yes
`allow_default`	number	Whether default route override is allowed, `0`=no, `1`=yes

`status`

Value	Meaning
`0`	Not enabled
`1`	Connection successful
`2`	Connecting (requesting network configuration)
`3`	Authentication required
`4`	Access denied
`5`	Network does not exist
`6`	Port error
`7`	Client version too old
`8`	Disconnected

2.5.10.2 Add/Modify ZeroTier Network

Request

```
{
  "core": "yruo_zerotier",
  "function": "set",
  "values": [
    {
      "base": "yruo_zerotier",
      "index": 1,
      "value": {
        "name": "Network1",
        "enable": 1,
        "network_id": "a1b2c3d4e5f6a7b8",
        "allow_managed": 1,
        "allow_global": 0,
        "allow_default": 0
      }
    }
  ]
}
```

> Add and modify use the same `set` interface, distinguished by `index` to target the record. When `allow_managed=0`, `ipaddr` must also be passed.

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count

Field	Type	Description
`templet_params`	array	Error parameters

2.5.10.3 Delete ZeroTier Network

Request

```
{
  "core": "yruo_zerotier",
  "function": "del",
  "values": [
    {
      "base": "yruo_zerotier",
      "index": 1,
      "value": {}
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains an error structure (as above).

2.5.10.4 Reset Node ID

Regenerate the device's ZeroTier Node ID.

Request

```
{
  "core": "yruo_zerotier",
  "function": "update_node_id",
  "values": [{}]
}
```

Response

On success `status=0`. After the operation completes, call the GET interface again to obtain the new `node_id` and `node_id_status`.

2.5.11 Certificate Management

> The `/cgi` interface uses `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.
> The certificate upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with `multipart/form-data` format.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.5.11.1 Get Certificate Status

Request

```
{
  "core": "yruo_vpn_certificate",
  "function": "get",
  "values": [{ "base": "yruo_vpn_certificate" }]
}
```

Response

```
{
  "id": 29,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_vpn_openvpn_client_certificate",
          "index": 1,
          "value": {
            "openvpn_client_ca": "",
            "openvpn_cient_crt": "",
            "openvpn_client_key": "",
            "openvpn_server_ta": "",
            "openvpn_server_tls": "",
            "openvpn_server_static": "",
            "openvpn_client_pkcs12": ""
          }
        },
        {
          "type": "yruo_vpn_openvn_server_certificate",
          "index": 1,
          "value": {
            "openvpn_server_ca": "",
            "openvpn_server_crt": "",
```

```
            "openvpn_server_key": "",
            "openvpn_server_dh": "",
            "openvpn_server_ta": "",
            "openvpn_server_tls": "",
            "openvpn_server_crl": "",
            "openvpn_server_static": ""
          }
        },
        {
          "type": "yruo_vpn_ipsec_certificate",
          "index": 1,
          "value": {
            "ipsec_ca": "",
            "ipsec_local_crt": "",
            "ipsec_remote_crt": "",
            "ipsec_key": "",
            "ipsec_crl": ""
          }
        },
        {
          "type": "yruo_vpn_ipsec_server_certificate",
          "index": 1,
          "value": {
            "ipsec_server_ca": "",
            "ipsec_server_local_crt": "",
            "ipsec_server_key": "",
            "ipsec_server_crl": ""
          }
        }
      ]
    }
  ]
}
```


> Each field value is the filename of the currently uploaded certificate file; empty string when not uploaded.
>
> Note: in `yruo_vpn_openvn_server_certificate`, `openvn` is the actual type name returned by the interface (missing the letter `p`); the documentation records it per the actual value.

Field Descriptions

`yruo_vpn_openvpn_client_certificate` value fields (OpenVPN client certificates)

Field	Type	Description
`openvpn_client_ca`	string	CA certificate file name
`openvpn_cient_crt`	string	Client public key file name (field name contains a typo, missing the letter `l`)
`openvpn_client_key`	string	Client private key file name
`openvpn_server_ta`	string	TLS Auth key file name
`openvpn_server_tls`	string	TLS Crypt key file name

Field	Type	Description
`openvpn_server_static`	string	Pre-shared key file name
`openvpn_client_pkcs12`	string	PKCS12 certificate package file name

`yruo_vpn_openvn_server_certificate` value fields (OpenVPN server certificates)

Field	Type	Description
`openvpn_server_ca`	string	CA certificate file name
`openvpn_server_crt`	string	Server public key file name
`openvpn_server_key`	string	Server private key file name
`openvpn_server_dh`	string	DH parameter file name
`openvpn_server_ta`	string	TLS Auth key file name
`openvpn_server_tls`	string	TLS Crypt key file name
`openvpn_server_crl`	string	CRL certificate revocation list file name
`openvpn_server_static`	string	Pre-shared key file name

`yruo_vpn_ipsec_certificate` value fields (IPsec client certificates)

Field	Type	Description
`ipsec_ca`	string	CA certificate file name
`ipsec_local_crt`	string	Client public key file name
`ipsec_remote_crt`	string	Server public key file name
`ipsec_key`	string	Client private key file name
`ipsec_crl`	string	CRL certificate revocation list file name

`yruo_vpn_ipsec_server_certificate` value fields (IPsec server certificates)

Field	Type	Description
`ipsec_server_ca`	string	CA certificate file name
`ipsec_server_local_crt`	string	Server public key file name
`ipsec_server_key`	string	Server private key file name
`ipsec_server_crl`	string	CRL certificate revocation list file name

2.5.11.2 Upload Certificate File

All certificates are uploaded via `POST /cgi-bin/file-import`, `Content-Type: multipart/form-data`.

Request fields

Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	`type=<filetype>&file=<file>`, see table below
`size`	File size (bytes)
`file`	File binary content

OpenVPN client certificates (`filetype=openvpn_client`)

`file` value	Accepted format	Description
`CA Certificate`	`.crt`	CA certificate
`Public Key`	`.crt`	Client public key
`Private Key`	`.key`	Client private key
`TA`	`.key`	TLS Auth key
`TA-crypt`	`.key`	TLS Crypt key
`Pre-shared Key`	`.key`	Pre-shared key
`PKCS12`	`.p12`	PKCS12 certificate package

OpenVPN server certificates (`filetype=openvpn_server`)

`file` value	Accepted format	Description
`CA Certificate`	`.crt`	CA certificate
`Public Key`	`.crt`	Server public key
`Private Key`	`.key`	Server private key
`DH`	`.pem`	DH parameter
`TA`	`.key`	TLS Auth key
`TA-crypt`	`.key`	TLS Crypt key
`CRL`	`.crl`	Certificate revocation list
`Pre-shared Key`	`.key`	Pre-shared key

IPsec client certificates (`filetype=ipsec`)

`file` value	Accepted format	Description
`CA Certificate`	`.crt`	CA certificate
`Client Public Key`	`.crt`	Client public key
`Server Public Key`	`.crt`	Server public key
`Private Key`	`.key`	Client private key
`CRL`	`.crl`	Certificate revocation list

IPsec server certificates (`filetype=ipsec_server`)

`file` value	Accepted format	Description
`CA Certificate`	`.crt`	CA certificate
`Client Public Key`	`.crt`	Server public key
`Private Key`	`.key`	Server private key
`CRL`	`.crl`	Certificate revocation list

Request example

Upload OpenVPN client CA certificate:

```
POST /cgi-bin/file-import
Content-Type: multipart/form-data; boundary=----FormBoundary

------FormBoundary
Content-Disposition: form-data; name="sessionid"

<td>
------FormBoundary
Content-Disposition: form-data; name="filename"

type=openvpn_client&file=CA Certificate
------FormBoundary
Content-Disposition: form-data; name="size"

1234
------FormBoundary
Content-Disposition: form-data; name="file"; filename="ca.crt"
Content-Type: application/octet-stream

<file binary content>
------FormBoundary--
```

Success response


```
{ "size": 1234, "checksum": "abc123", "filename": "openvpn_client-CA
Certificate.crt" }
```


Failure response


```
{ "templet_code": 10000001, "params_count": 0, "templet_params": [] }
```

2.6 IP Passthrough

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.6.1 Get IP Passthrough Configuration

Request


```
{
  "core": "yruo_ip_passthrough",
  "function": "get",
  "values": [{ "base": "yruo_ip_passthrough" }]
}
```


Response


```
{
  "id": 11,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
```

```
          "type": "yruo_ip_passthrough",
          "index": 0,
          "value": {
            "enable": 0,
            "mode": 0,
            "mac": "",
            "interface": "SIM1-APN1",
            "if_array": ["SIM1-APN1"]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_ip_passthrough` value fields

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`interface`	string	Outgoing interface name, selected from `if_array`
`mode`	number	Passthrough mode, `0`=DHCPS-Fixed, `1`=DHCPS-Dynamic
`mac`	string	Bound client MAC address (effective when `mode=0`)
`if_array`	string[]	List of available outgoing interface names (read-only)

2.6.2 Save IP Passthrough Configuration

Request

```
{
  "core": "yruo_ip_passthrough",
  "function": "set",
  "values": [
    {
      "base": "yruo_ip_passthrough",
      "type": "yruo_ip_passthrough",
      "index": 0,
      "value": {
        "enable": 1,
        "interface": "SIM1-APN1",
        "mode": 0,
        "mac": "AA:BB:CC:DD:EE:FF"
      }
    }
  ]
}
```

> `mac` only needs to be passed when `mode=0` (DHCPS-Fixed).

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.7 Routing

2.7.1 Static Routing

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.7.1.1 Get Static Routing Configuration

Request

```
{
  "core": "yruo_routestatic",
  "function": "get",
  "values": [{ "base": "yruo_routestatic" }]
}
```

Response

```
{
  "id": 7,
  "model": "UR35",
  "status": 0,
```

```
  "result": [
    {
      "get": [
        {
          "type": "yruo_routestatic_ifarray",
          "index": "qwerty",
          "value": {
            "if_array": ["vlan4", "vlan3", "vlan2", "dmvpn", "SIM1-APN1",
"WAN/LAN5", "Bridge0"],
            "track_id_array": []
          }
        },
        {
          "type": "yruo_routestatic",
          "index": "KBxSp65C9XOC6p9zf01X1Sz",
          "value": {
            "dst": "2400:3200::1",
            "mask": "128",
            "gw": "2409:8934:20b3:2c2d:e586:e237:533b:c7de",
            "ifn": "SIM1-APN1",
            "distance": 1,
            "track_id": 0,
            "shadow": 0
          }
        },
        {
          "type": "yruo_routestatic",
          "index": "85LODqn8d1e4yDbNTyj3b8N79j9B3ye34347L",
          "value": {
            "dst": "0.0.0.0",
            "mask": "0.0.0.0",
            "gw": "192.168.50.1",
            "ifn": "WAN/LAN5",
            "distance": 1,
            "track_id": 0,
            "shadow": 1
          }
        }
        // more yruo_routestatic entries omitted
      ]
    }
  ]
}
```


The response contains two types:

- `yruo_routestatic_ifarray`: a fixed single entry providing the list of available interfaces (index is the fixed string `"qwerty"`)
- `yruo_routestatic`: one per static route; index is a random string ID

Field Descriptions

`yruo_routestatic_ifarray` **value fields**

Field	Type	Description
`if_array`	string[]	List of available outgoing interface names
`track_id_array`	array	List of available Track IDs

`yruo_routestatic` **value fields**

Field	Type	Description
`dst`	string	Destination network address (IPv4 or IPv6)
`mask`	string	Netmask (IPv4 format like `255.255.255.0`, IPv6 prefix length like `"128"`)
`gw`	string	Gateway address (IPv4 or IPv6)
`ifn`	string	Outgoing interface name, `"NULL"` means unspecified
`distance`	number	Administrative distance, range 1–255
`track_id`	number	Track ID, `0` means unbound
`shadow`	number	Whether it is a shadow route (system auto-generated), `0`=no, `1`=yes; shadow routes cannot be edited or deleted

> At least one of `gw` and `ifn` must be filled. Routes with `shadow=1` are auto-generated by the system; the front end shows them read-only and they cannot be modified or deleted.

2.7.1.2 Save Static Routing Configuration

Request

```
{
  "core": "yruo_routestatic",
  "function": "set",
  "values": [
    {
      "base": "yruo_routestatic",
      "type": "yruo_routestatic",
      "index": "KBxSp65C9XOC6p9zf01X1Sz",
      "value": {
        "dst": "192.168.100.0",
        "mask": "255.255.255.0",
        "gw": "192.168.50.1",
        "ifn": "WAN/LAN5",
        "distance": 1
      }
    }
  ]
```

```
}
```

> When adding a route, `index` is a newly generated random string; on modification, pass the original `index`.

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.7.2 Policy Routing

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.7.2.1 Get Policy Routing List

Request

```
{
  "core": "yruo_policy_route",
  "function": "get",
  "values": []
}
```

Response

```
{
  "id": 8,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_policy_route",
          "index": 1,
          "value": {
            "priority": 1,
            "src": "192.168.1.0/24",
            "interface": "SIM1-APN1",
            "dest": "10.0.0.0/8"
          }
        }
        // more entries omitted
      ]
    }
  ]
}
```

> In the actual device response, `result[0].get` may be an empty array (when there are no policy routes). Supports up to 20 records.

Field Descriptions

`yruo_policy_route` value fields

Field	Type	Description
`priority`	number	Priority, range 1–255, must be unique within the same device
`src`	string	Source subnet (IPv4 CIDR format, optional)
`interface`	string	Outgoing interface name
`dest`	string	Destination subnet (IPv4 CIDR format, required)

2.7.2.2 Get Available Interface List

Request

```
{
  "core": "yruo_policy_route",
  "function": "get_interface_list",
  "values": []
}
```

Response

```
{
  "id": 9,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "values": ["SIM1-APN1", "WAN/LAN5"]
    }
  ]
}
```

> The response data is in the `result[0].values` array (not `result[0].get`), which is the list of available outgoing interface names.

2.7.2.3 Save Policy Routing Configuration

Request

```
{
  "core": "yruo_policy_route",
  "function": "set",
  "values": [
    {
      "base": "yruo_policy_route",
      "type": "yruo_policy_route",
      "index": 1,
      "value": {
        "priority": 1,
        "src": "192.168.1.0/24",
        "interface": "SIM1-APN1",
        "dest": "10.0.0.0/8"
      }
    }
  ]
}
```

> On add, `index` is a newly assigned number; on modification, pass the original `index`. `src` is an optional field; when not filled it matches all source addresses.

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.7.3 RIP

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.7.3.1 Get RIP Configuration

Request

```
{
  "core": "yruo_routerip",
  "function": "get",
  "values": [{ "base": "yruo_routerip" }]
}
```

Response

```
{
  "id": 10,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_routerip",
          "index": 0,
          "value": {
            "enable": 1,
            "update": 30,
            "timeout": 180,
            "garbage": 120,
            "version": 2,
            "originate": 0,
            "metric": 1,
            "redi_conn": 0,
            "metric_conn": 0,
```

```
            "redi_static": 0,
            "metric_static": 0,
            "redi_ospf": 0,
            "metric_ospf": 0,
            "show_adv": 0,
            "ifn_array": ["vlan4", "vlan3", "vlan2", "Cellular 0", "WAN/LAN5",
"Bridge0"],
            "distance_mgt": [],
            "metric_mgt": [],
            "filter": [],
            "passive_if": [],
            "neighbor": [],
            "interface": [
              {
                "id": "svxg3c99yo7rC48J4ClTjzGP099cHj",
                "sver": 0,
                "rver": 0,
                "split_horizon": 1,
                "auth_mode": 0,
                "ifn": "vlan4"
              }
            ],
            "network": []
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`yruo_routerip` **value main fields**

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`update`	number	Update timer (seconds), range 5–2147483647, default 30
`timeout`	number	Timeout timer (seconds), range 5–2147483647, default 180
`garbage`	number	Garbage collection timer (seconds), range 5–2147483647, default 120
`version`	number	RIP version, see enum table
`originate`	number	Whether to advertise the default route, `0`=no, `1`=yes
`metric`	number	Default metric, range 1–16
`redi_conn`	number	Whether to redistribute connected routes, `0`=no, `1`=yes
`metric_conn`	number	Connected route redistribution metric, range 0–16 (effective when `redi_conn=1`)

Field	Type	Description
`redi_static`	number	Whether to redistribute static routes, `0`=no, `1`=yes
`metric_static`	number	Static route redistribution metric, range 0–16 (effective when `redi_static=1`)
`redi_ospf`	number	Whether to redistribute OSPF routes, `0`=no, `1`=yes
`metric_ospf`	number	OSPF route redistribution metric, range 0–16 (effective when `redi_ospf=1`)
`show_adv`	number	Advanced options expand state (UI collapse control), `0`=collapsed, `1`=expanded
`ifn_array`	string[]	List of available interface names (read-only)
`distance_mgt`	object[]	Administrative distance list, see sub-field descriptions
`metric_mgt`	object[]	Metric management list, see sub-field descriptions
`filter`	object[]	Route filter list, see sub-field descriptions
`passive_if`	object[]	Passive interface list, see sub-field descriptions
`interface`	object[]	Interface configuration list, see sub-field descriptions
`neighbor`	object[]	Neighbor list, see sub-field descriptions
`network`	object[]	Network list, see sub-field descriptions

`version`

Value	Meaning
`0`	Default
`1`	v1
`2`	v2

`distance_mgt` sub-fields

Field	Type	Description
`distance`	number	Administrative distance, range 1–255
`ip`	string	Target network IP (IPv4)
`mask`	string	Netmask
`acl`	string	ACL name

`metric_mgt` sub-fields

Field	Type	Description
`metric`	number	Metric, range 0–16
`dir`	number	Direction, `0`=in, `1`=out
`ifn`	string	Interface name
`acl`	string	ACL name

`filter` sub-fields

Field	Type	Description
`ptype`	number	Filter list type, `0`=access-list, `1`=prefix-list
`name`	string	List name
`dir`	number	Direction, `0`=in, `1`=out
`ifn`	string	Interface name (optional)

`passive_if` sub-fields

Field	Type	Description
`ifn`	string	Passive interface name

`interface` sub-fields

Field	Type	Description
`id`	string	Interface configuration unique identifier (read-only, generated by the device)
`ifn`	string	Interface name
`sver`	number	Send version, `0`=default, `1`=v1, `2`=v2
`rver`	number	Receive version, `0`=default, `1`=v1, `2`=v2
`split_horizon`	number	Split horizon, `1`=enabled, `0`=disabled
`auth_mode`	number	Authentication mode, `0`=none, `1`=MD5, `2`=plaintext
`auth_string`	string	Authentication key (mutually exclusive with `auth_keychain`, SET only)
`auth_keychain`	string	Authentication keychain name (mutually exclusive with `auth_string`, SET only)

`neighbor` sub-fields

Field	Type	Description
`ip`	string	Neighbor IP address (IPv4)

`network` **sub-fields**

Field	Type	Description
`ip`	string	Network IP address (IPv4)
`mask`	string	Netmask

2.7.3.2 Save RIP Configuration

Request

```
{
  "core": "yruo_routerip",
  "function": "set",
  "values": [
    {
      "base": "yruo_routerip",
      "type": "yruo_routerip",
      "index": 0,
      "value": {
        "enable": 1,
        "update": 30,
        "version": 2
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.7.4 OSPF

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.7.4.1 Get OSPF Configuration

Request

```
{
  "core": "yruo_routeospf",
  "function": "get",
  "values": [{ "base": "yruo_routeospf" }]
}
```

Response

```
{
  "id": 11,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_routeospf",
          "index": 0,
          "value": {
            "enable": 1,
            "router_id": "192.168.2.2",
            "abr_type": 3,
            "rfc_cpt": 0,
            "opaque": 0,
            "spf_delay": 0,
            "spf_holdtime": 50,
            "spf_max": 5000,
            "ref_bandwidth": 100,
            "redist_def_always": 0,
            "redist_def_metric": 0,
            "interface_adv_display": 0,
            "area_adv_display": 0,
            "redistribute_adv_display": 0,
```

```
            "ifn_array": ["vlan4", "vlan3", "vlan2", "Cellular 0", "WAN/LAN5",
"Bridge0"],
            "passive_if": [],
            "interface_base": [],
            "interface_adv": [],
            "network": [],
            "neighbor": [],
            "area": [],
            "area_range": [],
            "area_filter": [],
            "area_vlink": [],
            "redistribute": [],
            "dist_mgt": []
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`yruo_routeospf` **value main fields**

Field	Type	Description
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`router_id`	string	Router ID (IPv4 format)
`abr_type`	number	ABR type, see enum table
`rfc_cpt`	number	RFC compatibility mode, `0`=disabled, `1`=enabled
`opaque`	number	Opaque LSA, `0`=disabled, `1`=enabled
`spf_delay`	number	SPF delay (milliseconds), range 0–600000
`spf_holdtime`	number	SPF hold time (milliseconds), range 0–600000
`spf_max`	number	SPF maximum wait time (milliseconds), range 0–600000
`ref_bandwidth`	number	Reference bandwidth (Mbit), range 1–4294967
`redist_def_always`	number	Always redistribute the default route, `0`=no, `1`=yes
`redist_def_metric`	number	Default route redistribution metric, range 0–16777214
`redist_def_type`	number	Default route redistribution type, `0`=none, `1`=type1, `2`=type2

Field	Type	Description
`interface_adv_display`	number	Interface advanced options expand state (UI collapse control), `0`=collapsed, `1`=expanded
`area_adv_display`	number	Area advanced options expand state (UI collapse control), `0`=collapsed, `1`=expanded
`redistribute_adv_display`	number	Redistribution advanced options expand state (UI collapse control), `0`=collapsed, `1`=expanded
`ifn_array`	string[]	List of available interface names (read-only)
`interface_base`	object[]	Interface base configuration list, see sub-field descriptions
`interface_adv`	object[]	Interface advanced configuration list, see sub-field descriptions
`passive_if`	object[]	Passive interface list, see sub-field descriptions
`network`	object[]	Network list, see sub-field descriptions
`neighbor`	object[]	Neighbor list, see sub-field descriptions
`area`	object[]	Area list, see sub-field descriptions
`area_range`	object[]	Area summary list, see sub-field descriptions
`area_filter`	object[]	Area filter list, see sub-field descriptions
`area_vlink`	object[]	Virtual link list, see sub-field descriptions
`redistribute`	object[]	Redistribution list, see sub-field descriptions
`dist_mgt`	object[]	Administrative distance list, see sub-field descriptions

`abr_type`

Value	Meaning
`1`	standard
`2`	ibm
`3`	cisco
`4`	shortcut

`interface_base` sub-fields

Field	Type	Description
`ifn`	string	Interface name

Field	Type	Description
`hello`	number	Hello interval (seconds), range 1–65535
`dead`	number	Dead interval (seconds), range 1–65535
`retr_int`	number	Retransmit interval (seconds), range 3–65535
`trans_delay`	number	Transmit delay (seconds), range 1–65535

`interface_adv` sub-fields

Field	Type	Description
`ifn`	string	Interface name
`type`	number	Interface type, see enum table
`cost`	number	Link cost, range 1–65535
`priority`	number	DR priority, range 0–255
`auth`	number	Authentication method, `-1`=none, `0`=simple, `1`=md5
`key_id`	number	Key ID, range 1–255 (effective when `auth=1`)
`key`	string	Authentication key (SET only, not returned by GET)

`interface_adv.type`

Value	Meaning
`1`	point-to-point
`2`	broadcast
`3`	non-broadcast
`4`	point-to-multipoint

`passive_if` sub-fields

Field	Type	Description
`ifn`	string	Passive interface name

`network` sub-fields

Field	Type	Description
`ip`	string	Network IP address (IPv4)
`mask`	string	Netmask
`area_id`	string	Area ID (IPv4 format or integer string)

`neighbor` sub-fields

Field	Type	Description
`ip`	string	Neighbor IP address (IPv4)
`priority`	number	Neighbor priority, range 0–255
`poll`	number	Poll interval (seconds), range 1–65535

`area` sub-fields

Field	Type	Description
`area_id`	string	Area ID (IPv4 format or integer string)
`area_type`	number	Area type, see enum table
`nosum`	number	No summary (effective for stub/nssa), `0`=no, `1`=yes
`auth_type`	number	Authentication type, `0`=none, `1`=simple, `2`=md5

`area.area_type`

Value	Meaning
`0`	None
`1`	stub
`2`	nssa-translate-candidata
`3`	nssa-translate-never
`4`	nssa-translate-always

`area_range` sub-fields

Field	Type	Description
`area_id`	string	Area ID
`ip`	string	Summary network IP (IPv4)
`mask`	string	Netmask
`not_adver`	number	Do not advertise, `0`=no, `1`=yes
`cost`	number	Cost, range 0–16777215

`area_filter` sub-fields

Field	Type	Description
`area_id`	string	Area ID

Field	Type	Description
`filter_type`	number	Filter type, `1`=import, `2`=export, `3`=filter-in, `4`=filter-out
`acl`	string	ACL name

`area_vlink` sub-fields

Field	Type	Description
`area_id`	string	Area ID
`vl_peer`	string	Virtual link peer Router ID (IPv4)
`auth_type`	number	Authentication type, `0`=simple, `1`=md5
`key_id`	number	Key ID, range 1–255 (effective when `auth_type=1`)
`key`	string	Authentication key (SET only, not returned by GET)
`hello`	number	Hello interval (seconds), range 1–65535
`dead`	number	Dead interval (seconds), range 1–65535
`retr_int`	number	Retransmit interval (seconds), range 3–65535
`trans_delay`	number	Transmit delay (seconds), range 1–65535

`redistribute` sub-fields

Field	Type	Description
`redist_type`	number	Redistribution source, `1`=connected, `2`=static, `3`=rip
`metric`	number	Metric, range 0–16777214
`metric_type`	number	Metric type, `1`=type1, `2`=type2
`map`	string	Route Map name

`dist_mgt` sub-fields

Field	Type	Description
`area_type`	number	Route type, `1`=intra-area, `2`=inter-area, `3`=external
`area_dist`	number	Administrative distance, range 1–255

2.7.4.2 Save OSPF Configuration

Request

```
{
  "core": "yruo_routeospf",
  "function": "set",
  "values": [
    {
      "base": "yruo_routeospf",
      "type": "yruo_routeospf",
      "index": 0,
      "value": {
        "enable": 1,
        "router_id": "192.168.2.2",
        "abr_type": 3
      }
    }
  ]
}
```

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.7.5 Route Filtering

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.7.5.1 Get Route Filtering Configuration

Request

```
{
  "core": "yruo_routefilter",
  "function": "get",
  "values": [{ "base": "yruo_routefilter" }]
}
```

Response

```
{
  "id": 12,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_routefilter_acl",
          "index": 1,
          "value": {
            "list_type": 0,
            "name": "acl1",
            "atype": 1,
            "any_match": 0,
            "ip": "192.168.1.0",
            "mask": "255.255.255.0"
          }
        },
        {
          "type": "yruo_routefilter_prefix",
          "index": 1,
          "value": {
            "list_type": 1,
            "pname": "prefix1",
            "seq": 10,
            "ptype": 1,
            "any_match": 0,
            "pip": "10.0.0.0",
            "pmask": "255.0.0.0",
            "ge": 8,
            "le": 24
          }
        }
      ]
    }
  ]
}
```

> In the actual device response, `result[0].get` may be an empty array (when there are no filter rules). The response contains two types, corresponding to ACL lists and prefix lists respectively.

The response contains two types:

- `yruo_routefilter_acl`: Access Control List (ACL) entries, up to 15
- `yruo_routefilter_prefix`: Prefix list entries, up to 5

Field Descriptions

`yruo_routefilter_acl` value fields

Field	Type	Description
`list_type`	number	List type, fixed at `0` (ACL)
`name`	string	ACL name
`atype`	number	Action, `0`=deny, `1`=permit
`any_match`	number	Match any, `0`=no, `1`=yes; when `1`, `ip`/`mask` are invalid
`ip`	string	Matching network IP (IPv4, required when `any_match=0`)
`mask`	string	Netmask (required when `any_match=0`)

`yruo_routefilter_prefix` value fields

Field	Type	Description
`list_type`	number	List type, fixed at `1` (prefix list)
`pname`	string	Prefix list name
`seq`	number	Sequence number, range 1–4294967295
`ptype`	number	Action, `0`=deny, `1`=permit
`any_match`	number	Match any, `0`=no, `1`=yes; when `1`, `pip`/`pmask`/`ge`/`le` are invalid
`pip`	string	Matching network IP (IPv4, required when `any_match=0`)
`pmask`	string	Netmask (required when `any_match=0`)
`ge`	number	Prefix length lower bound, range 0–32 (must be greater than the mask bit count)
`le`	number	Prefix length upper bound, range 0–32 (must be greater than or equal to `ge`)

2.7.5.2 Save Route Filtering Configuration

Request

Save an ACL entry:

```
{
  "core": "yruo_routefilter",
  "function": "set",
  "values": [
    {
      "base": "yruo_routefilter",
      "type": "yruo_routefilter_acl",
      "index": 1,
      "value": {
        "list_type": 0,
        "name": "acl1",
        "atype": 1,
        "any_match": 0,
        "ip": "192.168.1.0",
        "mask": "255.255.255.0"
      }
    }
  ]
}
```

Save a prefix list entry:

```
{
  "core": "yruo_routefilter",
  "function": "set",
  "values": [
    {
      "base": "yruo_routefilter",
      "type": "yruo_routefilter_prefix",
      "index": 1,
      "value": {
        "list_type": 1,
        "pname": "prefix1",
        "seq": 10,
        "ptype": 1,
        "any_match": 0,
        "pip": "10.0.0.0",
        "pmask": "255.0.0.0",
        "ge": 8,
        "le": 24
      }
    }
  ]
}
```

> The `type` field distinguishes ACL (`yruo_routefilter_acl`) from prefix list (`yruo_routefilter_prefix`).

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.8 VRRP

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.8.1 Get VRRP Configuration

Request

```
{
  "core": "yruo_vrrp",
  "function": "get",
  "values": [{ "base": "vrrp" }]
}
```

Response

```
{
  "id": 15,
  "model": "UR35",
  "status": 0,
  "result": [
    {
```

```
      "interface": ["Bridge0"],
      "trackid": [" "],
      "get": [
        {
          "type": "vrrp",
          "index": 1,
          "value": {
            "status": 99,
            "enable": 0,
            "virtual_router_id": 1,
            "interface": "Bridge0",
            "virtual_ip": "",
            "priority": 100,
            "advertise": 1,
            "preemption": 0,
            "dest_main": "8.8.8.8",
            "dest_second": "223.5.5.5",
            "icmp_period": 300,
            "icmp_interval": 5,
            "icmp_timeout": 3,
            "icmp_times": 3
          }
        }
      ]
    }
  ]
}
```


`result[0]` top-level fields

Field	Type	Description
`interface`	string[]	List of available interface names
`trackid`	string[]	List of available Track IDs

`vrrp` value field descriptions

Field	Type	Description
`status`	number	Current VRRP status (read-only), known values in enum table
`enable`	number	Whether enabled, `0`=disabled, `1`=enabled
`virtual_router_id`	number	Virtual router ID, range 1–255
`interface`	string	Bound interface name, selected from `result[0].interface`
`virtual_ip`	string	Virtual IP address (IPv4)
`priority`	number	Priority, range 1–254
`advertise`	number	Advertise interval (seconds), range 1–255

Field	Type	Description
`preemption`	number	Whether preemption is enabled, `0`=disabled, `1`=enabled
`dest_main`	string	Primary probe target IP (IPv4)
`dest_second`	string	Secondary probe target IP (IPv4, optional)
`icmp_period`	number	ICMP check period (seconds), range 10–1800
`icmp_interval`	number	ICMP check interval (seconds), range 3–600
`icmp_timeout`	number	ICMP timeout (seconds), range 1–10
`icmp_times`	number	ICMP check count, range 1–10

`status` (known enum values, may be incomplete)

Value	Meaning
`99`	Unknown/Not enabled

2.8.2 Save VRRP Configuration

Request

```
{
  "core": "yruo_vrrp",
  "function": "set",
  "values": [
    {
      "base": "vrrp",
      "type": "vrrp",
      "index": 1,
      "value": {
        "enable": 1,
        "virtual_router_id": 1,
        "old_virtual_router_id": 1,
        "interface": "Bridge0",
        "old_interface": "Bridge0",
        "virtual_ip": "192.168.1.254",
        "priority": 100,
        "advertise": 1,
        "preemption": 0,
        "dest_main": "8.8.8.8",
        "dest_second": "223.5.5.5",
        "icmp_period": 300,
        "icmp_interval": 5,
        "icmp_timeout": 3,
        "icmp_times": 3
      }
    }
  ]
```

```
}
```

> `old_virtual_router_id` and `old_interface` are the original values before modification, used to identify the target record; both must be passed on SET.

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

2.9 DDNS

> All requests use `POST https://<device-ip>/cgi`, carrying session credentials via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request status, `0` indicates success

2.9.1 Get DDNS Configuration

Request

```
{
  "core": "yruo_ddns",
  "function": "get",
  "values": [{ "base": "yruo_ddns" }]
}
```

Response


```
{
  "id": 16,
  "model": "UR35",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_ddns",
          "index": 0,
          "value": {
            "if_array": [
              { "ifname": "SIM1-APN1", "ip": "10.224.145.147" },
              { "ifname": "WAN/LAN5", "ip": "192.168.50.3" },
              { "ifname": "AUTO" }
            ],
            "adv_display": 0,
            "other_display": 0,
            "status": 0,
            "base": [
              {
                "id": "ri95L9vogzsOePSN2O",
                "name": "33",
                "ifn": "FEO",
                "cust_interface": "SIM1-APN1",
                "servicetype": 0,
                "enabled": 0,
                "username": "22",
                "password": "222",
                "status": 0,
                "use_https": 1,
                "userid": "33",
                "app_ip": 0
              }
            ],
            "adv": [
              {
                "id": "85WEmE43381T5qtoTpq",
                "name": "33",
                "ck_ssl": 0,
                "use_https": 0,
                "wildcard": 0
              }
            ],
            "other": [
              {
                "id": "TKxf5fXQIBUuG585u89m",
                "name": "33",
                "period": 120,
                "verify_addr": 1,
                "fake_addr": 1,
                "ipv6": 0,
                "forced_update_interval": 43200,
```

```
                "ssl": 1
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_ddns` value main fields

Field	Type	Description
`if_array`	object[]	List of available interfaces, see sub-field descriptions
`adv_display`	number	Advanced options expand state (UI collapse control), `0`=collapsed, `1`=expanded
`other_display`	number	Other options expand state (UI collapse control), `0`=collapsed, `1`=expanded
`status`	number	Overall status (known enum values, may be incomplete)
`base`	object[]	Base configuration list, see sub-field descriptions
`adv`	object[]	Advanced configuration list, see sub-field descriptions
`other`	object[]	Other configuration list, see sub-field descriptions

`if_array` sub-fields

Field	Type	Description
`ifname`	string	Interface name
`ip`	string	Interface current IP address (optional; `AUTO` interface has no such field)

`base` sub-fields

Field	Type	Description
`id`	string	Unique identifier (read-only, generated by the device)
`name`	string	DDNS configuration name
`ifn`	string	Interface name (internal field, retained by the device on SET)
`cust_interface`	string	Selected outgoing interface, chosen from `if_array[].ifname`

Field	Type	Description
`servicetype`	number	DDNS service provider, see enum table
`enabled`	number	Whether enabled, `0`=disabled, `1`=enabled
`username`	string	Account username
`password`	string	Account password
`status`	number	Connection status, `1`=connected, other=not connected
`use_https`	number	Whether to use HTTPS, `0`=no, `1`=yes (not applicable when `servicetype=30`)
`userid`	string	User ID (not applicable when `servicetype=30`)
`app_ip`	number	Whether to append the local IP, `0`=no, `1`=yes (effective when `servicetype=29`)
`hostname`	string	Hostname (optional, may be empty when `servicetype=29`)
`server`	string	Custom server address (only effective when `servicetype=29`)
`path`	string	Custom update path (only effective when `servicetype=29`)

`servicetype`

Value	Meaning
`0`	DynDNS
`1`	FreeDNS
`2`	Zoneedit
`3`	No-IP
`4`	EasyDNS
`6`	3322
`10`	Dynsip
`11`	Sitelutions
`12`	DnsExit
`13`	ChangeIP
`15`	DHIS
`17`	DuckDNS
`18`	LoopiaDNS

Value	Meaning
`19`	Google
`20`	OVH
`29`	custom
`30`	Oray

`adv` sub-fields

Field	Type	Description
`id`	string	Unique identifier (read-only, generated by the device)
`name`	string	Associated `base[].name`
`ck_ssl`	number	Whether to check SSL, `0`=no, `1`=yes
`use_https`	number	Whether to use HTTPS, `0`=no, `1`=yes
`wildcard`	number	Whether wildcard is enabled, `0`=no, `1`=yes

`other` sub-fields

Field	Type	Description
`id`	string	Unique identifier (read-only, generated by the device)
`name`	string	Associated `base[].name`
`period`	number	Update period (seconds)
`verify_addr`	number	Whether to verify the address, `0`=no, `1`=yes
`fake_addr`	number	Whether to allow fake addresses, `0`=no, `1`=yes
`ipv6`	number	Whether IPv6 is enabled, `0`=no, `1`=yes
`forced_update_interval`	number	Forced update interval (seconds), range 121–2592000
`ssl`	number	Whether SSL is enabled, `0`=no, `1`=yes

2.9.2 Save DDNS Configuration

Request

```
{
  "core": "yruo_ddns",
  "function": "set",
  "values": [
```

```
    {
      "base": "yruo_ddns",
      "type": "yruo_ddns",
      "index": 0,
      "value": {
        "base": [
          {
            "id": "ri95L9vogzs0ePSN2O",
            "name": "myDDNS",
            "ifn": "FE0",
            "cust_interface": "SIM1-APN1",
            "servicetype": 0,
            "enabled": 1,
            "username": "user",
            "password": "pass",
            "use_https": 1,
            "userid": "uid",
            "app_ip": 0
          }
        ],
        "other": [
          {
            "id": "TKxf5fXQIBUuG585u89m",
            "name": "myDDNS",
            "action": 2,
            "forced_update_interval": 43200
          }
        ]
      }
    }
  ]
}
```

> `other[].action`: `1`=add (when no existing other record), `2`=modify (when an other record exists). `forced_update_interval` is stored in `other[]`; the front end copies it to `base[0]` for display after GET, and moves it back to `other[]` on SET.

Response

On success `status=0`, `result[0]` has no error fields. On failure `result[0]` contains:

Field	Type	Description
`templet_code`	number	Error code
`params_count`	number	Parameter count
`templet_params`	array	Error parameters

3. System

3.1 General

3.1.1 General

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with `multipart/form-data` format.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, with `application/x-www-form-urlencoded` format.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

Get General Settings

Request

```
{
  "id": 12,
  "function": "get",
  "core": "yruo_system",
  "values": [{ "base": "general" }]
}
```

Response

```
{
  "id": 12,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
```

```
      "get": [
        {
          "type": "general",
          "index": 0,
          "value": {
            "hostname": "ROUTER",
            "web_login_timeout": 1800,
            "https_certificates": "https.crt",
            "https_key": "https.key",
            "passwd_crypto": 1
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`hostname`	string	Hostname, 1–31 characters
`web_login_timeout`	integer	Web login timeout (seconds), range 100–3600
`https_certificates`	string	HTTPS certificate file name (read-only, file updated via upload interface)
`https_key`	string	HTTPS private key file name (read-only, file updated via upload interface)
`passwd_crypto`	integer	Plaintext password encryption, `0`=disabled, `1`=enabled

---

Save General Settings

Request

```
{
  "id": 13,
  "function": "set",
  "core": "yruo_system",
  "values": [
    {
      "base": "general",
      "type": "general",
      "index": 0,
      "value": {
        "hostname": "MY-ROUTER",
        "web_login_timeout": 1800,
        "passwd_crypto": 1
      }
    }
  ]
}
```

Response

```
{
  "id": 13,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Upload HTTPS Certificate

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
```

Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	`type=https&file=certificate`
`size`	File size (bytes)
`file`	Binary content of the `.crt` file

Response

Success:

```
{ "size": 1234, "checksum": "abc123", "filename": "https.crt" }
```

Failure:

```
{ "templet_code": 1, "params_count": 0, "templet_params": [] }
```

Upload HTTPS Private Key

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
```

Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	`type=https&file=key`

Field	Description
`size`	File size (bytes)
`file`	Binary content of the `.key` file

Response

Success:

```
{ "size": 512, "checksum": "def456", "filename": "https.key" }
```

Failure:

```
{ "templet_code": 1, "params_count": 0, "templet_params": [] }
```

Download HTTPS Certificate

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded

sessionid=<td>&type=https&file=certificate
```

Response

```
Content-Disposition: attachment; filename="https.crt"
Content-Type: application/octet-stream

<binary file content>
```

Download HTTPS Private Key

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded

sessionid=<td>&type=https&file=key
```

Response

```
Content-Disposition: attachment; filename="https.key"
Content-Type: application/octet-stream

<binary file content>
```

3.1.2 System Time

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Get System Time Settings

Request

```
{
  "id": 13,
  "function": "get",
  "core": "yruo_system",
  "values": [{ "base": "time" }]
}
```

Response

```
{
  "id": 13,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "time",
          "index": 0,
          "value": {
            "current_time": "2026-04-10 09:01:21 Friday",
            "timezone": "UTC Europe/London",
            "timezone_type": 1,
            "set_type": 0,
            "ntp_enable": 1,
            "ntp_address": "pool.ntp.org",
            "ntp_address_secondary": "",
            "update_interval": 0,
            "ntp_master": 0,
            "gps_enable": 0
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`current_time`	string	**Read-only**, current device time, format `yyyy-MM-dd HH:mm:ss Weekday`
`timezone`	string	Timezone, format `UTC<offset> <Region/City>`
`timezone_type`	integer	Daylight saving time, `0`=disabled, `1`=enabled
`set_type`	integer	Time synchronization method, see table below
`ntp_enable`	integer	NTP client enabled, `0`=disabled, `1`=enabled (automatically set to 1 when `set_type`=0)
`ntp_address`	string	Primary NTP server address, max 256 characters (effective when `set_type`=0)
`ntp_address_secondary`	string	Secondary NTP server address, max 256 characters (effective when `set_type`=0)
`update_interval`	integer	NTP synchronization interval
`ntp_master`	integer	NTP server mode, `0`=disabled, `1`=enabled
`gps_enable`	integer	**Read-only**, whether GPS is enabled, `0`=disabled, `1`=enabled (determined by the GPS configuration page)

set_type Enum Values

Value	Description
`0`	NTP sync (automatically sets `ntp_enable`=1)
`1`	Sync with browser (SET request must carry `set_date`)
`2`	Manual setting (SET request must carry `set_date`)
`3`	GPS sync (only available when GPS hardware is present)
`4`	Cellular network sync

Save System Time Settings (NTP Mode)

Request

```
{
  "id": 14,
  "function": "set",
  "core": "yruo_system",
  "values": [
    {
      "base": "time",
```

```
      "type": "time",
      "index": 0,
      "value": {
        "timezone": "UTC+8 Asia/Shanghai",
        "timezone_type": 0,
        "set_type": 0,
        "ntp_enable": 1,
        "ntp_address": "pool.ntp.org",
        "ntp_address_secondary": "cn.pool.ntp.org",
        "update_interval": 0,
        "ntp_master": 0
      }
    }
  ]
}
```

Save System Time Settings (Manual Mode)

When `set_type`=2 (manual) or `set_type`=1 (sync with browser), the `set_date` field must be additionally carried.

Request

```
{
  "id": 15,
  "function": "set",
  "core": "yruo_system",
  "values": [
    {
      "base": "time",
      "type": "time",
      "index": 0,
      "value": {
        "timezone": "UTC+8 Asia/Shanghai",
        "timezone_type": 0,
        "set_type": 2,
        "set_date": "2026.04.10-09:30:00"
      }
    }
  ]
}
```

Field	Type	Description
`set_date`	string	Target time for manual/browser sync, format `yyyy.MM.dd-HH:mm:ss`

Response


```
{
  "id": 15,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.1.3 Email

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Get Email Settings

Request


```
{
  "id": 14,
  "function": "get",
  "core": "yruo_system",
  "values": [{ "base": "email" }]
}
```


Response


```
{
  "id": 14,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "email",
          "index": 0,
          "value": {
            "enable": 0,
            "email_test_address": "li15305961130@gmail.com",
            "email_address": "",
            "username": "",
            "password": "",
            "smtp_server": "",
            "port": 25,
            "enable_tls": 2,
```

```
            "recipients": [],
            "smtp_enable": 0,
            "email_list": [
              {
                "id": "azH3KBtt4ru5OC27y51M4kO1uK5PeZUNah2oO79ZRKnO9rp9Rzv",
                "email": "li153O5961132@gmail.com",
                "desc": "li153O5961130@gmail.com"
              }
            ],
            "group_list": [
              {
                "gid": 23,
                "desc": "23",
                "emails": [
                  "li153O5961133@gmail.com"
                ]
              }
            ]
          }
        }
      ]
    }
  ]
}
```


Field Description

Field	Type	Description
`enable`	integer	Whether email sending is enabled, `0`=disabled, `1`=enabled
`email_address`	string	Sender email address, max 63 characters
`username`	string	SMTP authentication username, max 128 characters (defaults to `email_address` when not set)
`password`	string	SMTP authentication password (returns empty string in GET response, plaintext is sent on SET)
`smtp_server`	string	SMTP server address, max 127 characters
`port`	integer	SMTP port, range 1–65535
`enable_tls`	integer	Encryption method: `0`=None, `1`=TLS/SSL, `2`=STARTTLS
`email_test_address`	string	Test email recipient address, max 63 characters
`smtp_enable`	integer	SMTP enabled (reserved field)
`recipients`	array	Recipient list (reserved field, currently unused)
`email_list`	array	Email address list, up to 15 entries, see table below
`group_list`	array	Email group list, up to 15 entries, see table below

email_list Object Fields

Field	Type	Description
`id`	string	Unique identifier of the email entry
`email`	string	Email address, max 63 characters
`desc`	string	Description, max 127 characters

group_list Object Fields

Field	Type	Description
`gid`	integer	Email group ID, range 1–100, unique within the group
`desc`	string	Description, max 127 characters
`emails`	array	Array of email addresses in the group (references the email values in `email_list`)

Save Email Settings

Request


```
{
  "id": 15,
  "function": "set",
  "core": "yruo_system",
  "values": [
    {
      "base": "email",
      "type": "email",
      "index": 0,
      "value": {
        "enable": 1,
        "email_address": "sender@gmail.com",
        "username": "sender@gmail.com",
        "password": "mypassword",
        "smtp_server": "smtp.gmail.com",
        "port": 587,
        "enable_tls": 2,
        "email_test_address": "receiver@example.com",
        "email_list": [
          {
            "id": "azH3KBtt4ru5OC27y51M4kO1uK5PeZUNah2oO79ZRKnO9rp9Rzv",
            "email": "li15305961132@gmail.com",
            "desc": "contact1"
          }
        ],
        "group_list": [
          {
            "gid": 1,
            "desc": "alert-group",
```

```
            "emails": ["li15305961132@gmail.com"]
          }
        ]
      }
    }
  ]
}
```

> The `password` field always returns an empty string in GET responses; on SET, send the new plaintext password, or omit the field if unchanged.

Response

```
{
  "id": 15,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Send Test Email

Sends a test email to the address configured in `email_test_address`. Ensure email settings have been saved before calling.

Request

```
{
  "id": 16,
  "execute": 1,
  "core": "yruo_system",
  "function": "test_mail",
  "values": []
}
```

Response

Success:

```
{
  "id": 16,
  "status": 0,
  "result": [{ "test": 0 }]
}
```

Failure:

```
{
  "id": 16,
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 0,
      "templet_params": []
    }
  ]
}
```

3.1.4 Storage

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Core: `yruo_storage` (Note: this module uses an independent core, different from the `yruo_system` used by other system settings)

Get Storage Device Status

Request


```
{
  "id": 15,
  "function": "get",
  "core": "yruo_storage",
  "values": [{ "base": "tfcard" }]
}
```


Response


```
{
  "id": 15,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "tfcard",
          "index": 0,
          "value": {
            "state": 0,
            "total": "-"
          }
        },
        {
```

```
          "type": "ssd",
          "index": 0,
          "value": {
            "state": 0,
            "total": "-"
          }
        }
      ]
    }
  ]
}
```

> The response contains both a `tfcard` (TF card) and an `ssd` (solid-state drive) record, distinguished by the `type` field. SSD is only shown on specific models (model number contains `7`).

Field Description

Field	Type	Description
`state`	integer	Storage device status, see table below (known enum values, may be incomplete)
`total`	string	Total storage capacity (e.g. `"16GB"`); `"-"` when not inserted

state Enum Values (known enum values, may be incomplete)

Value	Description
`0`	Not inserted / unavailable
`1`	Normal (formatable)
`2`	Inserted, formatable

Format Storage Device

Request

Format TF card:

```
{
  "id": 16,
  "execute": 1,
  "core": "yruo_storage",
  "function": "format",
  "values": [{ "base": "tfcard" }]
}
```

Format SSD:

```
{
  "id": 16,
  "execute": 1,
  "core": "yruo_storage",
  "function": "format",
  "values": [{ "base": "ssd" }]
}
```

> The format operation is only allowed when `state` is `1` or `2`.

Response

```
{
  "id": 16,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Query Format Progress

The format operation runs asynchronously; after triggering, poll the progress until complete (recommended interval 60 seconds).

Request

```
{
  "id": 17,
  "execute": 1,
  "core": "yruo_storage",
  "function": "format_progress",
  "values": [{ "base": "tfcard" }]
}
```

Response

```
{
  "id": 17,
  "status": 0,
  "result": [
    {
      "format_progress": 0
    }
  ]
}
```

Field	Type	Description
`result[0].format_progress`	integer	Format progress: `0`=completed

> `format_progress` is located directly at the top level of `result[0]`, not inside the `get[]` array.

3.2 Phone & SMS

3.2.1 Phone

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

Get Phone List

Request

```
{
  "id": 20,
  "function": "get",
  "core": "yruo_phone",
  "values": [{ "base": "yruo_phone" }]
}
```

Response

```
{
  "id": 20,
  "model": "UR35",
  "pn": "L04EU2211110CN00000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_phone",
          "index": 0,
```

```
          "value": {
            "number_list": [
              {
                "id": "DYM4416HBy94imZ811SXJc4EH6404ZN857nUD",
                "number": "222222222233"
              },
              {
                "id": "679nbuf20XkfHL6vys522Bf98vlbl04pmoM1",
                "number": "1111111111122"
              }
            ],
            "group_list": [
              {
                "gid": 3,
                "desc": "3",
                "numbers": [
                  "222222222222",
                  "11111111111"
                ]
              }
            ]
          }
        ]
      }
    ]
}
```


Field Description

Field	Type	Description
`number_list`	array	Phone number list, up to 15 entries, see table below
`group_list`	array	Phone group list, up to 15 entries, see table below

number_list Object Fields

Field	Type	Description
`id`	string	Unique identifier of the number entry
`number`	string	Phone number, max 32 characters
`desc`	string	Description, max 127 characters

group_list Object Fields

Field	Type	Description
`gid`	integer	Phone group ID, range 1–100
`desc`	string	Description, max 127 characters

Field	Type	Description
`numbers`	array	Array of phone number strings in the group (references the `number` values in `number_list`)

Save Phone List

Request

```
{
  "id": 21,
  "function": "set",
  "core": "yruo_phone",
  "values": [
    {
      "base": "yruo_phone",
      "type": "yruo_phone",
      "index": 0,
      "value": {
        "number_list": [
          {
            "id": "DYM4416HBy94imZ811SXJc4EH6404ZN857nUD",
            "number": "13800138000",
            "desc": "contact1"
          }
        ],
        "group_list": [
          {
            "gid": 1,
            "desc": "alert-group",
            "numbers": ["13800138000"]
          }
        ]
      }
    }
  ]
}
```

> When adding a new number entry, do not include `id`; when editing, include the original `id`.
> A phone number cannot be deleted while it is referenced by a phone group.

Response

```
{
  "id": 21,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.2.2 SMS

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

Get SMS Settings

Request

```
{
  "id": 21,
  "function": "get",
  "core": "yruo_sms",
  "values": [{ "base": "yruo_sms_setting" }]
}
```

Response

```
{
  "id": 21,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_sms_setting",
          "index": 0,
          "value": {
            "phone_groups": [3],
            "sms_mode": 0,
            "sms_auth_passwd": "",
            "sms_ctl_enable": 0,
            "sms_auth_type": 0,
```

```
            "sms_auth_group": 0
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`phone_groups`	array	**Read-only**, list of currently available phone group IDs (used to populate the `sms_auth_group` dropdown options)
`sms_mode`	integer	SMS mode: `0`=PDU, `1`=TEXT
`sms_ctl_enable`	integer	Whether SMS control is enabled: `0`=disabled, `1`=enabled
`sms_auth_type`	integer	Authentication method (effective when `sms_ctl_enable`=1): `0`=authorized numbers only, `1`=number+password
`sms_auth_passwd`	string	Authentication password, max 63 characters (effective when `sms_auth_type`=1)
`sms_auth_group`	integer	Authorized phone group ID (effective when `sms_ctl_enable`=1)

Save SMS Settings

Request

```
{
  "id": 22,
  "function": "set",
  "core": "yruo_sms",
  "values": [
    {
      "base": "yruo_sms_setting",
      "type": "yruo_sms_setting",
      "index": 0,
      "value": {
        "sms_mode": 0,
        "sms_ctl_enable": 1,
        "sms_auth_type": 1,
        "sms_auth_passwd": "mypassword",
        "sms_auth_group": 3
      }
    }
  ]
}
```

Response


```
{
  "id": 22,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Send SMS

Request


```
{
  "id": 23,
  "function": "send",
  "core": "yruo_sms",
  "values": [
    {
      "base": "yruo_sms",
      "index": 1,
      "value": {
        "destination": "13800138000",
        "content": "Hello World"
      }
    }
  ]
}
```


Field	Type	Description
`destination`	string	Destination phone number
`content`	string	SMS content, max 180 characters

Response

Success:


```
{
  "id": 23,
  "status": 0,
  "result": []
}
```


Failure:

```
{
  "id": 23,
  "status": 0,
  "result": [
    { "templet_code": 1, "params_count": 0, "templet_params": [] }
  ]
}
```

Query Inbox

Request

```
{
  "id": 24,
  "function": "query_inbox",
  "core": "yruo_sms",
  "values": [
    {
      "base": "query_inbox",
      "limit": 10,
      "start": 0,
      "language": "en",
      "key": "time",
      "order": 0,
      "start_date": "",
      "end_date": "",
      "from": ""
    }
  ]
}
```

Field	Type	Description
`limit`	integer	Number of entries per page
`start`	integer	Starting offset
`language`	string	UI language identifier
`key`	string	Sort field, currently `"time"`
`order`	integer	Sort direction, `0`=descending
`start_date`	string	Start time filter, format `yyyy-mm-dd hh:ii`, empty string means no limit
`end_date`	string	End time filter, format `yyyy-mm-dd hh:ii`, empty string means no limit
`from`	string	Sender number filter, empty string means no limit

Response


```
{
  "id": 24,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "timezone": "UTC Europe/London",
      "count": 24,
      "get": [
        {
          "type": "sms_inbox",
          "index": 1,
          "value": {
            "from": "10086",
            "time": "2026-04-06 08:05:26",
            "content": "[Password Service Reminder]..."
          }
        },
        {
          "type": "sms_inbox",
          "index": 2,
          "value": {
            "from": "10086",
            "time": "2026-04-06 08:05:26",
            "content": "A friendly reminder:..."
          }
        }
        // ...more entries
      ]
    }
  ]
}
```


result[0] Top-level Fields

Field	Type	Description
`timezone`	string	Current device timezone
`count`	integer	Total number of SMS messages matching the criteria (used for pagination)

sms_inbox Field Description

Field	Type	Description
`from`	string	Sender number
`time`	string	Receive time, format `yyyy-MM-dd HH:mm:ss`

Field	Type	Description
`content`	string	SMS content

Query Outbox

Request

```
{
  "id": 25,
  "function": "query_outbox",
  "core": "yruo_sms",
  "values": [
    {
      "base": "query_outbox",
      "limit": 10,
      "start": 0,
      "language": "en",
      "key": "time",
      "order": 0,
      "start_date": "",
      "end_date": "",
      "from": ""
    }
  ]
}
```

> In the outbox context, the `from` field represents the filter for the destination number (recipient).

Response

```
{
  "id": 25,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "timezone": "UTC Europe/London",
      "count": 0,
      "get": []
    }
  ]
}
```

sms_outbox Field Description

Field	Type	Description
`from`	string	Destination number (recipient)
`time`	string	Send time, format `yyyy-MM-dd HH:mm:ss`
`content`	string	SMS content
`status`	string	Send status

Clear Inbox

Request

```
{
  "id": 26,
  "function": "clear_inbox",
  "core": "yruo_sms",
  "values": [{ "base": "clear_inbox" }]
}
```

Response

```
{
  "id": 26,
  "status": 0,
  "result": [{ "timezone": "UTC Europe/London", "count": 0, "get": [] }]
}
```

Clear Outbox

Request

```
{
  "id": 27,
  "function": "clear_outbox",
  "core": "yruo_sms",
  "values": [{ "base": "clear_outbox" }]
}
```

Response

```
{
  "id": 27,
  "status": 0,
  "result": [{ "timezone": "UTC Europe/London", "count": 0, "get": [] }]
}
```

3.3 User Management

3.3.1 Account

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

---

Get Account Information

Request

```
{
  "id": 24,
  "function": "get",
  "core": "yruo_usermanagement",
  "values": [{ "base": "security" }]
}
```

Response

```
{
  "id": 24,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "security",
          "index": "cfg00000",
          "value": {
            "username": "admin",
            "permission": 4
          }
```

```
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`username`	string	Username of the currently logged-in account
`permission`	integer	Permission level, `4` =administrator (known enum value, may be incomplete)

> The password field is not returned in the GET response.

Modify Account (Username / Password)

Request

```
{
  "id": 25,
  "function": "set",
  "core": "yruo_usermanagement",
  "values": [
    {
      "base": "security",
      "type": "security",
      "index": "cfg00000",
      "value": {
        "username": "admin",
        "old_password": "oldpassword",
        "new_password": "newpass123",
        "confirm_password": "newpass123"
      }
    }
  ]
}
```

Field	Type	Required	Description
`username`	string	Yes	Username, 1–31 characters, must not use system-reserved names
`old_password`	string	Yes	Current password
`new_password`	string	Yes	New password, 5–31 characters, must contain both letters and numbers, and cannot be the same as the old password
`confirm_password`	string	Yes	Confirm new password, must match `new_password`

> After saving successfully, the device triggers a re-login flow (the current session becomes invalid).

> The following usernames are system-reserved and cannot be used: `root`, `support`, `adm`, `daemon`, `ftp`, `network`, `nobody`, `dnsmasq`, `mail`, `audio`, `www-data`, `users`, `nogroup`

Response

```
{
  "id": 25,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.3.2 User Management

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

Get User List

Request

```
{
  "id": 26,
  "function": "get",
  "core": "yruo_usermanagement",
  "values": [{ "base": "user_list" }]
}
```

Response


```
{
  "id": 26,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "user_list",
          "index": "cfg00001",
          "value": {
            "username": "admin2",
            "permission": 0
          }
        }
      ]
    }
  ]
}
```


Field Description

Field	Type	Description
`username`	string	Username
`permission`	integer	Permission level, `0`=read-only, `1`=operator

> The password field is not returned in the GET response. The administrator account (`permission: 4`) is managed via the Account page (3.3.1) and is not included in this list.

> Up to 5 users can be added.

Add / Modify User

Request


```
{
  "id": 27,
  "function": "set",
  "core": "yruo_usermanagement",
  "values": [
    {
      "base": "user_list",
      "type": "user_list",
      "index": "cfg00001",
      "value": {
        "old_username": "admin2",
        "username": "admin2",
```

```
        "password": "newpass123",
        "permission": 1
      }
    }
  ]
}
```

Field	Type	Required	Description
`old_username`	string	Yes (when editing)	Original username, used to identify the user being edited; same as `username` when creating a new user
`username`	string	Yes	Username, max 31 characters, must not use system-reserved names
`password`	string	Yes	Password, 5–31 characters, must contain both letters and numbers; when editing, send `"******"` to keep the password unchanged
`permission`	integer	Yes	Permission level, `0`=read-only, `1`=operator

> The following usernames are system-reserved and cannot be used: `root`, `support`, `adm`, `daemon`, `ftp`, `network`, `nobody`, `dnsmasq`, `mail`, `audio`, `www-data`, `users`, `nogroup`

Response

```
{
  "id": 27,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Delete User

Request

```
{
  "id": 28,
  "function": "del",
  "core": "yruo_usermanagement",
  "values": [
    {
      "base": "user_list",
      "type": "user_list",
      "index": "cfg00001",
      "value": {
        "username": "admin2"
      }
```

```
    }
  ]
}
```

Response

```
{
  "id": 28,
  "model": "UR35",
  "pn": "L04EU2211110CN00000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.4 AAA

3.4.1 RADIUS

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

Get RADIUS Configuration

Request

```
{
  "id": 28,
  "function": "get",
  "core": "yruo_aaa",
  "values": [{ "base": "radius" }]
}
```

Response


```
{
  "id": 28,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "radius",
          "index": 0,
          "value": {
            "enable": 0
          }
        }
      ]
    }
  ]
}
```


Field Description

Field	Type	Description
`enable`	integer	Whether enabled, `0`=disabled, `1`=enabled
`server_ip`	string	RADIUS server address (IP or domain), max 127 characters
`port`	integer	Server port, 1–65535, default `1812`

> `server_ip` and `port` may not be returned in the response when `enable=0`. The password field is not returned in GET responses.

Save RADIUS Configuration

Request


```
{
  "id": 29,
  "function": "set",
  "core": "yruo_aaa",
  "values": [
    {
      "base": "radius",
      "type": "radius",
      "index": 0,
      "value": {
        "enable": 1,
        "server_ip": "192.168.1.100",
        "port": 1812,
```

```
        "password": "secret"
      }
    }
  ]
}
```

Field	Type	Required	Description
`enable`	integer	Yes	`0`=disabled, `1`=enabled
`server_ip`	string	Required when enabled	RADIUS server address, max 127 characters
`port`	integer	Required when enabled	Server port, 1–65535
`password`	string	Required when enabled	Shared secret, max 63 characters

Response

```
{
  "id": 29,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.4.2 TACACS+

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

Get TACACS+ Configuration

Request

```
{
  "id": 29,
  "function": "get",
  "core": "yruo_aaa",
  "values": [{ "base": "tacplus" }]
}
```

Response

```
{
  "id": 29,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "tacplus",
          "index": 0,
          "value": {
            "enable": 0
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`enable`	integer	Whether enabled, `0`=disabled, `1`=enabled
`server_ip`	string	TACACS+ server address (IP or domain), max 127 characters
`port`	integer	Server port, 1–65535, default `49`

> `server_ip` and `port` may not be returned in the response when `enable=0`. The password field is not returned in GET responses.

Save TACACS+ Configuration

Request

```
{
  "id": 30,
  "function": "set",
  "core": "yruo_aaa",
  "values": [
    {
      "base": "tacplus",
      "type": "tacplus",
      "index": 0,
      "value": {
        "enable": 1,
        "server_ip": "192.168.1.100",
        "port": 49,
        "password": "secret"
      }
    }
  ]
}
```

Field	Type	Required	Description
`enable`	integer	Yes	`0`=disabled, `1`=enabled
`server_ip`	string	Required when enabled	TACACS+ server address, max 127 characters
`port`	integer	Required when enabled	Server port, 1–65535
`password`	string	Required when enabled	Shared secret, max 63 characters

Response

```
{
  "id": 30,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.4.3 LDAP

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

Get LDAP Configuration

Request

```
{
  "id": 30,
  "function": "get",
  "core": "yruo_aaa",
  "values": [{ "base": "ldap" }]
}
```

Response

```
{
  "id": 30,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "ldap",
          "index": 0,
          "value": {
            "enable": 0,
            "server_ip": "",
            "port": 389,
            "base_dn": "",
            "security": 0,
```

```
            "username": "",
            "password": ""
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`enable`	integer	Whether enabled, `0`=disabled, `1`=enabled
`server_ip`	string	LDAP server address (IP or domain), max 127 characters
`port`	integer	Server port, 1–65535, default `389`
`base_dn`	string	Base DN, max 63 characters
`security`	integer	Encryption method, see table below
`username`	string	Bind username (Bind DN), max 63 characters
`password`	string	Bind password, GET returns an empty string, SET sends plaintext, max 63 characters

`security` **Enum Values**

Value	Meaning
`0`	None (no encryption)
`1`	TLS
`2`	SSL

Save LDAP Configuration

Request

```
{
  "id": 31,
  "function": "set",
  "core": "yruo_aaa",
  "values": [
    {
      "base": "ldap",
      "type": "ldap",
      "index": 0,
      "value": {
        "enable": 1,
        "server_ip": "ldap.example.com",
```

```
        "port": 389,
        "base_dn": "dc=example,dc=com",
        "security": 0,
        "username": "cn=admin,dc=example,dc=com",
        "password": "secret"
      }
    }
  ]
}
```

Field	Type	Required	Description
`enable`	integer	Yes	`0`=disabled, `1`=enabled
`server_ip`	string	Required when enabled	LDAP server address, max 127 characters
`port`	integer	Required when enabled	Server port, 1–65535
`base_dn`	string	Required when enabled	Base DN, max 63 characters
`security`	integer	Required when enabled	Encryption method, `0`=None / `1`=TLS / `2`=SSL
`username`	string	Required when enabled	Bind username, max 63 characters
`password`	string	Required when `username` is non-empty	Bind password, max 63 characters

Response

```
{
  "id": 31,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.4.4 Authentication

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

---

Get Authentication Priority Configuration

Request

```
{
  "id": 31,
  "function": "get",
  "core": "yruo_aaa",
  "values": [{ "base": "authentication" }]
}
```

Response

```
{
  "id": 31,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "authentication",
          "index": 0,
          "value": { "service": "Console", "priority_1": 0, "priority_2": 0,
"priority_3": 0 }
        },
        {
          "type": "authentication",
          "index": 1,
          "value": { "service": "Web", "priority_1": 0, "priority_2": 0,
"priority_3": 0 }
        },
        {
          "type": "authentication",
          "index": 2,
```

```
          "value": { "service": "Telnet", "priority_1": 0, "priority_2": 0,
"priority_3": 0 }
        },
        {
          "type": "authentication",
          "index": 3,
          "value": { "service": "SSH", "priority_1": 0, "priority_2": 0,
"priority_3": 0 }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`service`	string	Service name (read-only), `Console` / `Web` / `Telnet` / `SSH`
`priority_1`	integer	First-priority authentication method, see table below
`priority_2`	integer	Second-priority authentication method; not effective when `priority_1=0`
`priority_3`	integer	Third-priority authentication method; not effective when `priority_2=0`

Priority Enum Values

Value	Meaning
`0`	None (this priority not used)
`1`	Local (local authentication)
`2`	RADIUS
`3`	TACACS+
`4`	LDAP

> When `priority_1=0`, `priority_2` and `priority_3` are forced to `0`; when `priority_2=0`, `priority_3` is forced to `0`.

Save Authentication Priority Configuration

Request

```
{
  "id": 32,
  "function": "set",
  "core": "yruo_aaa",
  "values": [
```

```
    {
      "base": "authentication",
      "type": "authentication",
      "index": 1,
      "value": {
        "priority_1": 1,
        "priority_2": 2,
        "priority_3": 0
      }
    }
  ]
}
```

> Each SET only needs to submit the entries that changed; the `service` field is read-only and does not need to be submitted.

Response

```
{
  "id": 32,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.5 Device Management

3.5.1 Auto Provision

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

Get Auto Provision Status

Request

```
{
  "id": 35,
  "function": "get",
  "core": "yruo_cloud",
  "values": [{ "base": "auto_provision" }]
}
```

Response

```
{
  "id": 35,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "auto_provision",
          "index": 0,
          "value": {
            "enable": 1,
            "status": 1
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`enable`	integer	Whether auto provision is enabled, `0`=disabled, `1`=enabled
`status`	integer	Current status (read-only), see table below
`err_msg`	string	Error message, only present when `status=2`

`status` Enum Values

Value	Color	Meaning
`0`	Gray	Not enabled
`1`	Red	Connection failed

Value	Color	Meaning
`2`	Red	Fetch failed (also returns `err_msg`)
`3`	Green	Provisioning
`4`	Green	Provisioned
`5`	Green	Connecting (frontend polls every 2 seconds)

Save Auto Provision

Request

```
{
  "id": 36,
  "function": "set",
  "core": "yruo_cloud",
  "values": [
    {
      "base": "auto_provision",
      "type": "auto_provision",
      "index": 0,
      "value": {
        "enable": 1
      }
    }
  ]
}
```

Field	Type	Required	Description
`enable`	integer	Yes	`0`=disabled, `1`=enabled

Response

```
{
  "id": 36,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.5.2 Device Management

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

---

Get Device Management Configuration

Request

```
{
  "id": 36,
  "function": "get",
  "core": "yruo_cloud",
  "values": [{ "base": "cloud_manage" }]
}
```

Response

```
{
  "id": 36,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "cloud_manage",
          "index": 0,
          "value": {
            "action": 0,
            "dm_type": 1,
            "status": 200,
            "active_srv": "",
            "cloud_url": "",
```

```
            "stun_enable": 0,
            "authcode": "",
            "method": 0
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`action`	integer	Whether connection is enabled, `0`=disabled, `1`=enabled
`dm_type`	integer	Platform type, see table below
`status`	integer	Connection status (read-only), see table below
`active_srv`	string	Activation server address, max 127 characters; only used when `dm_type=0`
`method`	integer	Activation method, `0`=authorization code, `1`=username+password; only used when `dm_type=0`
`authcode`	string	Authorization code, max 64 characters; only used when `dm_type=0` and `method=0`
`ysid`	string	Username, max 64 characters; only used when `dm_type=0` and `method=1`
`cloud_url`	string	Reserved field, GET returns an empty string
`stun_enable`	integer	Reserved field, GET returns `0`

> `passwd` (password) is only sent in SET and not returned in GET responses; only used when `dm_type=0` and `method=1`, max 64 characters.
>
> `dh_addr` (server address, IP or domain, max 128 characters) is only used when `dm_type=2`, and may only be returned by GET in that scenario.

`dm_type` Enum Values

Value	Meaning
`0`	DeviceHub V1
`1`	Development platform
`2`	DeviceHub V2

`status` Enum Values

Value	Color	Meaning
`200`	Gray	Not connected
`201`	Green	Connected
`202`	Green	Connecting (frontend polls every 2 seconds)
`203`	Gray	Disconnected (frontend polls every 2 seconds)
`204`	Red	Download failed

Save Device Management Configuration

Request

```
{
  "id": 37,
  "function": "set",
  "core": "yruo_cloud",
  "values": [
    {
      "base": "cloud_manage",
      "type": "cloud_manage",
      "index": 0,
      "value": {
        "action": 1,
        "dm_type": 0,
        "active_srv": "activate.milesight.com",
        "method": 0,
        "authcode": "XXXX-XXXX-XXXX-XXXX"
      }
    }
  ]
}
```

Field	Type	Required	Description
`action`	integer	Yes	`0`=disabled, `1`=enabled
`dm_type`	integer	Yes	Platform type
`active_srv`	string	Required when `dm_type=0`	Activation server, max 127 characters
`method`	integer	Required when `dm_type=0`	`0`=authorization code, `1`=username+password
`authcode`	string	Required when `dm_type=0` and `method=0`	Authorization code, max 64 characters

Field	Type	Required	Description
`ysid`	string	Required when `dm_type=0` and `method=1`	Username, max 64 characters
`passwd`	string	Required when `dm_type=0` and `method=1`	Password, max 64 characters
`dh_addr`	string	Required when `dm_type=2`	Server address (IP or domain), max 128 characters

Response

```
{
  "id": 37,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Disconnect

When the device is connected (`action=1`), you can send the following request to actively disconnect:

Request

```
{
  "id": 38,
  "function": "set",
  "core": "yruo_cloud",
  "values": [
    {
      "base": "cloud_manage",
      "index": 0,
      "value": { "action": 0 }
    }
  ]
}
```

Response

```
{
  "id": 38,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.5.3 Milesight VPN

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

---

Get Milesight VPN Configuration

Request

```
{
  "id": 37,
  "function": "get",
  "core": "yruo_urvpn",
  "values": [{ "base": "urvpn_manage" }]
}
```

Response

```
{
  "id": 37,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
```

```
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "urvpn_manage",
          "index": 0,
          "value": {
            "server": "",
            "port": 18443,
            "authcode": "",
            "device_name": "",
            "status": 300,
            "localip": "--",
            "remoteip": "--",
            "duration": 0
          }
        }
      ]
    }
  ]
}
```


Field Description

Configuration Fields (only editable when `status=300`)

Field	Type	Description
`server`	string	VPN server address, max 127 characters
`port`	integer	Server port, 1–65535, default `18443`
`authcode`	string	Authorization code, max 64 characters
`device_name`	string	Device name, max 64 characters

Status Fields (read-only)

Field	Type	Description
`status`	integer	Connection status, see table below
`localip`	string	Local VPN IP address, `"--"` when not connected
`remoteip`	string	Remote VPN IP address, `"--"` when not connected
`duration`	integer	Connection duration (seconds), `0` means not connected

`status` **Enum Values** (known enum values, may be incomplete)

Value	Color	Meaning
`300`	Gray	Not connected

Value	Color	Meaning
`301`	Green	Connected
`302`	Green	Connecting
`303`	Red	Abnormal
`304`	Red	Abnormal
`305`	Red	Abnormal
`306`	Red	Abnormal

> `duration` frontend rendering format: `N days,HH:MM:SS`; displays `-` when the value is `0`.

Connect Milesight VPN

Configuration can be submitted and a connection initiated only when the current `status=300` (not connected):

Request

```
{
  "id": 38,
  "function": "set",
  "core": "yruo_urvpn",
  "values": [
    {
      "base": "urvpn_manage",
      "type": "urvpn_manage",
      "index": 0,
      "value": {
        "server": "vpn.example.com",
        "port": 18443,
        "authcode": "XXXX-XXXX-XXXX-XXXX",
        "device_name": "MyDevice"
      }
    }
  ]
}
```

Field	Type	Required	Description
`server`	string	Yes	VPN server address, max 127 characters
`port`	integer	Yes	Server port, 1–65535
`authcode`	string	Yes	Authorization code, max 64 characters
`device_name`	string	Yes	Device name, max 64 characters

Response


```
{
  "id": 38,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Disconnect Milesight VPN

When `status≠300` (connected or connecting), you can send the following request to disconnect:

Request


```
{
  "id": 39,
  "function": "set",
  "core": "yruo_urvpn",
  "values": [
    {
      "base": "urvpn_manage",
      "index": 0,
      "value": { "action": 0 }
    }
  ]
}
```


Response


```
{
  "id": 39,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.6 Events

3.6.1 Events

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

---

Query Event Log

Request

```
{
  "id": 38,
  "function": "get",
  "core": "yruo_events",
  "values": [
    {
      "base": "event_log",
      "limit": 10,
      "start": 0,
      "language": "en",
      "key": "time",
      "order": 0
    }
  ]
}
```

Field	Type	Description
`limit`	integer	Number of entries per page
`start`	integer	Starting offset
`language`	string	UI language identifier
`key`	string	Sort field, options: `"time"`, `"status"`, `"eventid"`, `"message"`
`order`	integer	Sort direction, `0` =descending, `1` =ascending

Response

```
{
  "id": 38,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
```

```
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "count": 3396,
      "get": [
        {
          "type": "event_log",
          "index": 1,
          "value": {
            "id": 3396,
            "status": 0,
            "eventid": 4003,
            "time": "2026-04-10 00:24:29",
            "message": "SIM1 Up."
          }
        },
        {
          "type": "event_log",
          "index": 2,
          "value": {
            "id": 3395,
            "status": 0,
            "eventid": 4004,
            "time": "2026-04-10 00:22:21",
            "message": "SIM1 Down."
          }
        }
        // ...more entries
      ]
    }
  ]
}
```


result[0] Top-level Fields

Field	Type	Description
`count`	integer	Total number of events matching the criteria (used for pagination)

event_log Field Description

Field	Type	Description
`id`	integer	Unique event ID
`status`	integer	Read status, `0`=unread (frontend displays in bold), `1`=read
`eventid`	integer	Event type ID
`time`	string	Event time, format `yyyy-MM-dd HH:mm:ss`
`message`	string	Event message content

Mark Events as Read

Request

Mark selected entries:

```
{
  "id": 39,
  "function": "set",
  "core": "yruo_events",
  "values": [
    {
      "base": "event_log",
      "index": 0,
      "value": {
        "ackids": [3396, 3395]
      }
    }
  ]
}
```

Mark all:

```
{
  "id": 39,
  "function": "set",
  "core": "yruo_events",
  "values": [
    {
      "base": "event_log",
      "index": 0,
      "value": {
        "ackids": ["all"]
      }
    }
  ]
}
```

Field	Type	Description
`ackids`	array	List of event `id`s to mark as read; send `["all"]` to mark all

Response

```
{
  "id": 39,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Delete Events

Request

Delete selected entries:

```
{
  "id": 40,
  "function": "delete",
  "core": "yruo_events",
  "values": [
    {
      "base": "event_log",
      "index": 0,
      "value": {
        "delids": [3396, 3395]
      }
    }
  ]
}
```

Delete all:

```
{
  "id": 40,
  "function": "delete",
  "core": "yruo_events",
  "values": [
    {
      "base": "event_log",
      "index": 0,
      "value": {
        "delids": ["all"]
      }
    }
  ]
}
```

Field	Type	Description
`delids`	array	List of event `id`s to delete; send `["all"]` to delete all

Response

```
{
  "id": 40,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.6.2 Events Setting

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

Get Events Setting

Request

```
{
  "id": 39,
  "function": "get",
  "core": "yruo_events",
  "values": [{ "base": "event_list" }]
}
```

Response

```
{
  "id": 39,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "enable": 1,
      "phone_groupid": [3],
      "email_groupid": [23],
      "select_groupid": 0,
      "select_emailgroupid": 0,
      "get": [
        {
          "type": "event_list",
          "index": 1,
          "value": {
```

```
            "eventid": 4019,
            "subject": "system-boot",
            "record": 1,
            "email": 0,
            "sms": 0,
            "snmp": 0
          }
        },
        {
          "type": "event_list",
          "index": 2,
          "value": {
            "eventid": 4020,
            "subject": "system-reboot",
            "record": 1,
            "email": 0,
            "sms": 0,
            "snmp": 0
          }
        }
        // ...more event entries
      ]
    }
  ]
}
```

result[0] Top-level Fields

Field	Type	Description
`enable`	integer	Master switch for the events feature, `0`=disabled, `1`=enabled
`phone_groupid`	array	List of available phone group IDs (read-only, used to populate SMS/phone group dropdown options)
`email_groupid`	array	List of available email group IDs (read-only, used to populate email group dropdown options)
`select_groupid`	integer	Currently selected phone group ID
`select_emailgroupid`	integer	Currently selected email group ID

event_list Field Description

Field	Type	Description
`eventid`	integer	Event type ID
`subject`	string	Event name identifier
`record`	integer	Whether to log, `0`=no, `1`=yes
`email`	integer	Whether to send email, `0`=no, `1`=yes

Field	Type	Description
`sms`	integer	Whether to send SMS, `0`=no, `1`=yes
`snmp`	integer	Whether to send SNMP Trap, `0`=no, `1`=yes

Known Event ID List

`eventid`	`subject`	Description
`4003`	`cellular-up`	Cellular connected
`4004`	`cellular-down`	Cellular disconnected
`4005`	`wan-up`	WAN connected
`4006`	`wan-down`	WAN disconnected
`4007`	`vpn-up`	VPN connected
`4008`	`vpn-down`	VPN disconnected
`4019`	`system-boot`	System boot
`4020`	`system-reboot`	System reboot
`4021`	`system-time-update`	System time update
`4022`	`cellular-data-clear`	Cellular data cleared
`4023`	`cellular-data-near-threshold`	Cellular data near threshold
`4024`	`cellular-data-excess-threshold`	Cellular data exceeds threshold
`4025`	`cellular-signal-weak`	Cellular signal weak
`4026`	`link-switching`	Link switching
`4027`	`wlan1-connected-ap`	WLAN1 connected to AP
`4028`	`wlan1-disconnected-ap`	WLAN1 disconnected from AP
`4029`	`wlan1-connected-client`	WLAN1 client connected
`4030`	`wlan1-disconnected-client`	WLAN1 client disconnected
`4031`	`wlan2-connected-ap`	WLAN2 connected to AP
`4032`	`wlan2-disconnected-ap`	WLAN2 disconnected from AP
`4033`	`wlan2-connected-client`	WLAN2 client connected
`4034`	`wlan2-disconnected-client`	WLAN2 client disconnected
`4035`	`cellular-signal-recovery`	Cellular signal recovery
`4036`	`lan-up`	LAN connected

eventid	subject	Description
4037	lan-down	LAN disconnected
4038	sim-switch	SIM card switch

Save Events Setting

Request

```
{
  "id": 40,
  "function": "set",
  "core": "yruo_events",
  "values": [
    {
      "base": "event_list",
      "index": 0,
      "value": {
        "enable": 1,
        "record": [4019, 4020, 4003, 4004],
        "email": [4003],
        "sms": [4004],
        "snmp": [],
        "phone_groupid": 3,
        "select_emailgroupid": 23
      }
    }
  ]
}
```

Field	Type	Required	Description
enable	integer	Yes	Master switch for the events feature, 0=disabled, 1=enabled
record	array	Yes	List of eventids to log
email	array	Yes	List of eventids to send email
sms	array	Yes	List of eventids to send SMS
snmp	array	Yes	List of eventids to send SNMP Trap
phone_groupid	integer	No	Selected phone group ID (for SMS/phone notifications)
select_emailgroupid	integer	No	Selected email group ID

> The SET request submits each notification type as an array of eventids (rather than returning 0/1 per entry as in GET), which is structurally asymmetric with GET.

Response


```
{
  "id": 40,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

Get MQTT Client Name List

Used to populate the MQTT client dropdown options in the event MQTT push configuration.

Request


```
{
  "id": 40,
  "function": "get_name_list",
  "core": "yruo_service_mqtt",
  "values": []
}
```


Response


```
{
  "id": 40,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_service_mqtt",
          "index": 1,
          "value": {
            "mqtt_name_list": ["123"]
          }
        }
      ]
    }
  ]
}
```

Field Description

Field	Type	Description
`mqtt_name_list`	array	List of configured MQTT client names (string array)

Get Event MQTT Push Configuration

Request

```
{
  "id": 41,
  "function": "get",
  "core": "yruo_events",
  "values": [{ "base": "mqtt_publish" }]
}
```

Response

```
{
  "id": 41,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "mqtt_publish",
          "index": 0,
          "value": {
            "enable": 0,
            "list": []
          }
        }
      ]
    }
  ]
}
```

mqtt_publish Field Description

Field	Type	Description
`enable`	integer	Master switch for MQTT push, `0`=disabled, `1`=enabled
`list`	array	MQTT push rule list, up to 25 entries, see table below

list Entry Fields

Field	Type	Description
`id`	integer	Rule ID
`event_list`	array	List of `eventid`s that trigger this rule
`mqtt_list`	array	List of target MQTT client names
`topic`	string	Publish topic, max 511 characters
`flag`	integer	Retain flag, `0`=do not retain, `1`=retain
`qos`	integer	QoS level, `0` / `1` / `2`

Save Event MQTT Push Configuration

Request

```
{
  "id": 42,
  "function": "set",
  "core": "yruo_events",
  "values": [
    {
      "type": "mqtt_publish",
      "base": "mqtt_publish",
      "index": 0,
      "value": {
        "enable": 1,
        "list": [
          {
            "id": 1,
            "event_list": [4019, 4020],
            "mqtt_list": ["123"],
            "topic": "event/system",
            "flag": 0,
            "qos": 1
          }
        ]
      }
    }
  ]
}
```

> The `topic` + `mqtt_list` combination must be unique across all rules; the frontend will report an error on duplicates.

Response

```
{
  "id": 42,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": []
}
```

3.7 Power Management

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> This interface is only supported by UR4X.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

3.7.1 Query Standby Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_power",
  "values": [
    { "base": "yruo_power" }
  ]
}
```

Response


```
{
  "id": 6,
  "model": "UR41",
  "pn": "L08EU0011111DE0000000000",
  "oem": "0000",
  "rtver": "41.0.0.7-a2",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_power",
          "index": 0,
          "value": {
            "email_list": [],
            "phone_list": [],
            "enable": 0,
            "standby_sms": 0,
            "standby_phone_group": "",
            "standby_sms_content": "",
            "standby_email": 0,
            "standby_email_group": "",
            "standby_email_content": "",
            "standby_do": 0,
            "standby_do_level": 0,
            "standby_do_duration": 100,
            "standby_pulse_init": 0,
            "standby_pulse_high": 100,
            "standby_pulse_low": 100,
            "standby_pulse_amount": 10,
            "time_wakeup": 0,
            "time_mode": 0,
            "repeat_hours": 0,
            "repeat_days": [7, 1, 2, 3, 4, 5, 6],
            "wakeup_hours": 0,
            "wakeup_minutes": 10,
            "standby_hours": 0,
            "standby_minutes": 50,
            "di": 0,
            "di_level": 0,
            "di_duration": 1,
            "di_mode": 0,
            "di_standby_duration": 100,
            "di_wakeup_time": 1,
            "cellular": 0,
            "cell_phone_group": "",
            "cell_sms_group": "",
            "cell_sms_content": "",
            "cell_wakeup_time": 1,
            "ethernet": 0,
            "eth_wakeup_time": 1,
            "serial": 0,
            "serial_wakeup_time": 1,
```

```
            "wakeup_sms": 0,
            "wakeup_phone_group": "",
            "wakeup_sms_content": "",
            "wakeup_email": 0,
            "wakeup_email_group": "",
            "wakeup_email_content": "",
            "wakeup_do": 0,
            "wakeup_do_level": 0,
            "wakeup_do_duration": 100,
            "wakeup_pulse_init": 0,
            "wakeup_pulse_high": 100,
            "wakeup_pulse_low": 100,
            "wakeup_pulse_amount": 10
          }
        }
      ]
    }
  ]
}
```

Field Description

`yruo_power` **Type (type=`yruo_power`, index=`0`)**

Basic Settings

Field	Type	Description
`enable`	integer	Global enable, `0`=disabled, `1`=enabled
`email_list`	array<string>	Available email group list, read-only, used for `standby_email_group` / `wakeup_email_group` dropdown options
`phone_list`	array<string>	Available phone/SMS group list, read-only, used for `standby_phone_group` / `cell_phone_group` / `cell_sms_group` / `wakeup_phone_group` dropdown options

Pre-standby Actions (each set of sub-fields takes effect when `standby_sms`=1 or `standby_email`=1 or `standby_do`=1)

Field	Type	Description
`standby_sms`	integer	Send SMS before standby, `0`=no, `1`=yes
`standby_phone_group`	string	SMS phone group, value from `phone_list`, effective when `standby_sms`=1
`standby_sms_content`	string	Pre-standby SMS content, effective when `standby_sms`=1
`standby_email`	integer	Send email before standby, `0`=no, `1`=yes

Field	Type	Description
`standby_email_group`	string	Email group, value from `email_list`, effective when `standby_email`=1
`standby_email_content`	string	Pre-standby email content, effective when `standby_email`=1
`standby_do`	integer	Trigger DO before standby, `0`=no, `1`=yes
`standby_do_level`	integer	DO mode, see enum table, effective when `standby_do`=1
`standby_do_duration`	integer	DO high/low level duration (milliseconds), range 1–8640000, effective when `standby_do_level` is `0` or `1`
`standby_pulse_init`	integer	DO pulse initial state, `0`=high level, `1`=low level, effective when `standby_do_level`=2
`standby_pulse_high`	integer	DO pulse high level duration (milliseconds), range 1–10000, effective when `standby_do_level`=2
`standby_pulse_low`	integer	DO pulse low level duration (milliseconds), range 1–10000, effective when `standby_do_level`=2
`standby_pulse_amount`	integer	DO pulse count, range 1–100, effective when `standby_do_level`=2

Wakeup Settings - Time Wakeup (sub-fields take effect when `time_wakeup`=1)

Field	Type	Description
`time_wakeup`	integer	Enable time wakeup, `0`=no, `1`=yes
`time_mode`	integer	Repeat mode, `0`=hourly cycle, `1`=daily cycle
`repeat_hours`	integer	Hourly repeat period, see enum table, effective when `time_mode`=0
`repeat_days`	array<integer>	Daily repeat period, elements see enum table, effective when `time_mode`=1
`wakeup_hours`	integer	Wakeup time - hour, range 0–23
`wakeup_minutes`	integer	Wakeup time - minute, range 0–59
`standby_hours`	integer	Standby time - hour, range 0–23
`standby_minutes`	integer	Standby time - minute, range 0–59

Wakeup Settings - DI Wakeup (sub-fields take effect when `di`=1)

Field	Type	Description
`di`	integer	Enable DI wakeup, `0`=no, `1`=yes
`di_level`	integer	DI trigger level, `0`=high level, `1`=low level
`di_duration`	integer	DI trigger duration (seconds), range 1–30
`di_mode`	integer	DI trigger mode, `0`=DI continuously triggers standby, `1`=scheduled wakeup
`di_standby_duration`	integer	DI mode standby duration (milliseconds), range 1–30000, effective when `di_mode`=0
`di_wakeup_time`	integer	DI mode wakeup duration (minutes), range 1–1080, effective when `di_mode`=1

Wakeup Settings - Cellular Wakeup (sub-fields take effect when `cellular`=1)

Field	Type	Description
`cellular`	integer	Enable cellular wakeup, `0`=no, `1`=yes
`cell_phone_group`	string	Cellular wakeup incoming-call number group, value from `phone_list`
`cell_sms_group`	string	Cellular wakeup SMS number group, value from `phone_list`
`cell_sms_content`	string	Cellular wakeup SMS keyword content, max 127 characters
`cell_wakeup_time`	integer	Cellular wakeup duration (minutes), range 1–1080

Wakeup Settings - Ethernet Wakeup (sub-fields take effect when `ethernet`=1)

Field	Type	Description
`ethernet`	integer	Enable Ethernet wakeup, `0`=no, `1`=yes
`eth_wakeup_time`	integer	Ethernet wakeup duration (minutes), range 1–1080

Wakeup Settings - Serial Wakeup (sub-fields take effect when `serial`=1)

Field	Type	Description
`serial`	integer	Enable serial wakeup, `0`=no, `1`=yes
`serial_wakeup_time`	integer	Serial wakeup duration (minutes), range 1–1080

Post-wakeup Actions (each set of sub-fields takes effect when `wakeup_sms`=1 or `wakeup_email`=1 or `wakeup_do`=1)

Field	Type	Description
`wakeup_sms`	integer	Send SMS after wakeup, `0`=no, `1`=yes
`wakeup_phone_group`	string	SMS phone group, value from `phone_list`, effective when `wakeup_sms`=1
`wakeup_sms_content`	string	Post-wakeup SMS content, effective when `wakeup_sms`=1
`wakeup_email`	integer	Send email after wakeup, `0`=no, `1`=yes
`wakeup_email_group`	string	Email group, value from `email_list`, effective when `wakeup_email`=1
`wakeup_email_content`	string	Post-wakeup email content, max 62 characters, effective when `wakeup_email`=1
`wakeup_do`	integer	Trigger DO after wakeup, `0`=no, `1`=yes
`wakeup_do_level`	integer	DO mode, see enum table, effective when `wakeup_do`=1
`wakeup_do_duration`	integer	DO high/low level duration (milliseconds), range 1–8640000, effective when `wakeup_do_level` is `0` or `1`
`wakeup_pulse_init`	integer	DO pulse initial state, `0`=high level, `1`=low level, effective when `wakeup_do_level`=2
`wakeup_pulse_high`	integer	DO pulse high level duration (milliseconds), range 1–10000, effective when `wakeup_do_level`=2
`wakeup_pulse_low`	integer	DO pulse low level duration (milliseconds), range 1–10000, effective when `wakeup_do_level`=2
`wakeup_pulse_amount`	integer	DO pulse count, range 1–100, effective when `wakeup_do_level`=2

Enum Values

`standby_do_level` / `wakeup_do_level`

Value	Meaning
`0`	High level
`1`	Low level
`2`	Pulse

`time_mode`

Value	Meaning
`0`	Hourly cycle

Value	Meaning
`1`	Daily cycle

`repeat_hours` (effective when `time_mode`=0)

Value	Meaning
`0`	Every 1 hour
`1`	Every 2 hours
`2`	Every 3 hours
`3`	Every 4 hours
`4`	Every 6 hours
`5`	Every 8 hours
`6`	Every 12 hours

`repeat_days` **Element Values** (effective when `time_mode`=1, is an integer array)

Value	Meaning
`1`	Monday
`2`	Tuesday
`3`	Wednesday
`4`	Thursday
`5`	Friday

| `6` | Saturday |
| `7` | Sunday |

3.7.2 Save Standby Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_power",
  "values": [
    {
      "base": "yruo_power",
      "type": "yruo_power",
      "index": 0,
      "value": {
        "enable": 1,
```

```
        "time_wakeup": 1,
        "time_mode": 1,
        "repeat_days": [1, 2, 3, 4, 5],
        "wakeup_hours": 8,
        "wakeup_minutes": 0,
        "standby_hours": 18,
        "standby_minutes": 0
      }
    }
  ]
}
```

The `value` object only needs to include the changed fields. Submittable fields are symmetric with the `value` object fields in the GET response; `email_list` and `phone_list` are read-only fields and cannot be submitted.

Response

Success: `status`=`0`, `result[0].get` is an empty array `[]`.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

4. Service

4.1 IO

4.1.1 Digital Input

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version

Field	Type	Description
`status`	number	Request execution status: `0` success

4.1.1.1 Get Digital Input Configuration

Request

```
{
  "id": 13,
  "execute": 1,
  "core": "yruo_io_input",
  "function": "get",
  "values": [{ "base": "yruo_io_input" }]
}
```

Response

```
{
  "id": 13,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_io_input_1",
          "index": 1,
          "value": {
            "email_list": ["23"],
            "phone_list": ["3"],
            "level": 0,
            "duration": 100,
            "action_sms": 0,
            "action_email": 0,
            "action_do_1": 0,
            "action_do_2": 0,
            "action_cellular_up": 0,
            "action_mqtt_forward": 0,
            "action_snmp_trap": 0,
            "io_status": 1,
            "enable": 0
          }
        },
        {
          "type": "yruo_io_input_2",
          "index": 2,
          "value": {
            "email_list": ["23"],
            "phone_list": ["3"],
```

```
            "level": 0,
            "duration": 100,
            "action_sms": 0,
            "action_email": 0,
            "action_do_1": 0,
            "action_do_2": 0,
            "action_cellular_up": 0,
            "action_mqtt_forward": 0,
            "action_snmp_trap": 0,
            "io_status": 0,
            "enable": 0
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

The response contains two records, corresponding to DI1 (`yruo_io_input_1`) and DI2 (`yruo_io_input_2`).

`value` **Object Fields**

Field	Type	Writable	Description
`enable`	number	✅	Whether enabled: `1` enabled, `0` disabled
`level`	number	✅	Trigger level mode, see enum table
`duration`	number	✅	Trigger duration (ms), range 1–10000; valid when `level ≠ 2`
`condition`	number	✅	Counter trigger condition, `0` =low→high, `1` =high→low; only valid when `level = 2` (inferred from code, may not appear in actual response)
`counter`	number	✅	Count threshold, range 1–100; only valid when `level = 2` (inferred from code, may not appear in actual response)
`action_sms`	number	✅	Send SMS on trigger: `1` enabled, `0` disabled (requires device to have cellular capability)
`action_email`	number	✅	Send email on trigger: `1` enabled, `0` disabled
`action_do_1`	number	✅	Link DO1 output on trigger: `1` enabled, `0` disabled

Field	Type	Writable	Description
`action_do_2`	number	✅	Link DO2 output on trigger: `1` enabled, `0` disabled
`action_cellular_up`	number	✅	Dial up on trigger: `1` enabled, `0` disabled (requires device to have cellular capability)
`action_mqtt_forward`	number	✅	MQTT forwarding on trigger: `1` enabled, `0` disabled
`action_snmp_trap`	number	✅	Send SNMP Trap on trigger: `1` enabled, `0` disabled
`phone`	string	✅	SMS recipient, selected from `phone_list`; valid when `action_sms = 1` (inferred from code, may not appear in actual response)
`sms_content`	string	✅	SMS content; valid when `action_sms = 1` (inferred from code, may not appear in actual response)
`email`	string	✅	Recipient email group, selected from `email_list`; valid when `action_email = 1` (inferred from code, may not appear in actual response)
`email_content`	string	✅	Email body, up to 62 characters; valid when `action_email = 1` (inferred from code, may not appear in actual response)
`forward_mqtt_list`	array	✅	MQTT forwarding rule list; valid when `action_mqtt_forward = 1`, see sub-table below (only DI1 UI provides this feature)
`email_list`	array	❌	List of available email group names, determined by system mailbox configuration, only used for frontend dropdown options
`phone_list`	array	❌	List of available phone group names, determined by system phone configuration, only used for frontend dropdown options
`io_status`	number	❌	Current hardware input status: `1` high level, `0` low level (read-only)

`level` Enum Values

Value	Description	Applicable
`0`	High level trigger	DI1 / DI2
`1`	Low level trigger	DI1 / DI2
`2`	Counter mode	DI1 / DI2
`4`	Switch mode	DI1 only

`forward_mqtt_list[]` Sub-fields

Field	Type	Description
`mqtt_name`	string	MQTT connection name, selected from the system MQTT service list
`topic`	string	Publish topic, up to 511 characters
`flag`	number	Retain flag: `1` retained, `0` not retained
`qos`	number	QoS level: `0` / `1` / `2`

4.1.1.2 Get MQTT Service Name List

Used to populate the MQTT forwarding connection name dropdown when the page loads.

Request

```
{
  "id": 14,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "get_name_list",
  "values": []
}
```

Response

```
{
  "id": 14,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_service_mqtt",
          "index": 1,
```

```
          "value": {
            "mqtt_name_list": ["123"]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

Field	Type	Description
`mqtt_name_list`	array<string>	Configured MQTT connection name list, used as connection name options for DI1 MQTT forwarding rules

---

4.1.1.3 Save Digital Input Configuration

Request

The SET request `values` contains only the changed entries, each corresponding to one DI channel.

```
{
  "id": 15,
  "execute": 1,
  "core": "yruo_io_input",
  "function": "set",
  "values": [
    {
      "base": "yruo_io_input",
      "type": "yruo_io_input_1",
      "index": 1,
      "value": {
        "enable": 1,
        "level": 0,
        "duration": 200,
        "action_email": 1,
        "email": "group1",
        "email_content": "DI1 triggered"
      }
    }
  ]
}
```

The fields that can be submitted in `value` refer to all writable (✅) fields in 4.1.1.1 Field Descriptions.

> At least one action must be checked (one of `action_sms`, `action_email`, `action_do_1`, `action_do_2`, `action_cellular_up`, `action_mqtt_forward`, `action_snmp_trap` must be `1`), otherwise the frontend validation fails.

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.1.2 Digital Output

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.1.2.1 Get Digital Output Configuration

Request

```
{
  "id": 15,
  "execute": 1,
  "core": "yruo_io_output",
  "function": "get",
  "values": [{ "base": "yruo_io_output" }]
}
```

Response


```
{
  "id": 15,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_io_output_1",
          "index": 1,
          "value": {
            "phone_list": ["3"],
            "level": 0,
            "duration": 100,
            "alarm_di_1": 0,
            "alarm_di_2": 0,
            "alarm_con_cell": 0,
            "alarm_dis_cell": 0,
            "io_status": 1,
            "enable": 0
          }
        },
        {
          "type": "yruo_io_output_2",
          "index": 2,
          "value": {
            "phone_list": ["3"],
            "level": 0,
            "duration": 100,
            "alarm_di_1": 0,
            "alarm_di_2": 0,
            "alarm_con_cell": 0,
            "alarm_dis_cell": 0,
            "io_status": 0,
            "enable": 0
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

The response contains two records, corresponding to DO1 (`yruo_io_output_1`) and DO2 (`yruo_io_output_2`).

`value` Object Fields

Field	Type	Writable	Description
`enable`	number	✅	Whether enabled: `1` enabled, `0` disabled
`level`	number	✅	Output mode, see enum table
`duration`	number	✅	Duration (ms), range 1–8640000; valid when `level = 0` or `1`
`init_status`	number	✅	Initial level: `0` high level, `1` low level; valid when `level = 2` or `3` (inferred from code, may not appear in actual response)
`phone`	string	✅	Phone group for incoming-call trigger, selected from `phone_list`; valid when `level = 3` (custom) (inferred from code, may not appear in actual response)
`duration_of_high`	number	✅	Pulse high level duration (ms), range 1–10000; valid when `level = 2` (inferred from code, may not appear in actual response)
`duration_of_low`	number	✅	Pulse low level duration (ms), range 1–10000; valid when `level = 2` (inferred from code, may not appear in actual response)
`amount_of_pulse`	number	✅	Number of pulses, range 1–100; valid when `level = 2` (inferred from code, may not appear in actual response)
`alarm_con_cell`	number	✅	Trigger on cellular dial-up: `1` enabled, `0` disabled
`alarm_dis_cell`	number	✅	Trigger on cellular disconnect: `1` enabled, `0` disabled
`alarm_di_1`	number	✅	Link output on DI1 trigger: `1` enabled, `0` disabled (not editable in current UI)
`alarm_di_2`	number	✅	Link output on DI2 trigger: `1` enabled, `0` disabled (not editable in current UI)
`phone_list`	array	❌	List of available phone group names, determined by system phone configuration, only used for frontend dropdown options
`io_status`	number	❌	Current hardware output status: `1` high level, `0` low level (read-only)

`level` Enum Values

Value	Description	Additional Fields
`0`	High level output	`duration`
`1`	Low level output	`duration`
`2`	Pulse output	`init_status`, `duration_of_high`, `duration_of_low`, `amount_of_pulse`
`3`	Custom (incoming call control)	`init_status`, `phone`

4.1.2.2 Save Digital Output Configuration

Request

The SET request `values` contains only the changed entries, each corresponding to one DO channel.

```
{
  "id": 16,
  "execute": 1,
  "core": "yruo_io_output",
  "function": "set",
  "values": [
    {
      "base": "yruo_io_output",
      "type": "yruo_io_output_1",
      "index": 1,
      "value": {
        "enable": 1,
        "level": 0,
        "duration": 500,
        "alarm_con_cell": 1
      }
    }
  ]
}
```

The fields that can be submitted in `value` refer to all writable (✅) fields in 4.1.2.1 Field Descriptions.

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.2 Serial Port

4.2.1 Serial Port 1

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.2.1.1 Get Serial Port 1 Configuration

Request

```
{
  "id": 16,
  "execute": 1,
  "core": "yruo_industrial_serial_1",
  "function": "get",
  "values": [{ "base": "yruo_industrial_serial_1" }]
}
```

Response


```
{
  "id": 16,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_industrial_serial_1",
          "index": 1,
          "value": {
            "serial_type": 1,
            "baudrate": 5,
            "data_bits": 1,
            "stop_bits": 1,
            "parity": 1,
            "flow_control": 0,
            "mode": 3,
            "enable": 1
          }
        }
      ]
    }
  ]
}
```


> The above is a response example for `mode=3` (Modbus Client), with the minimal field set. Under other modes, the corresponding extension fields are additionally included; see the field descriptions below.

Field Descriptions

Basic Fields (always present)

Field	Type	Writable	Description
`enable`	number	✅	Whether enabled: `1` enabled, `0` disabled
`serial_type`	number	✅	Serial port type, see enum table; the selectable range is determined by the hardware model
`baudrate`	number	✅	Baud rate, see enum table
`data_bits`	number	✅	Data bits: `1`=8bits, `2`=7bits
`stop_bits`	number	✅	Stop bits: `1`=1bit, `2`=2bits
`parity`	number	✅	Parity: `1`=None, `2`=Odd, `3`=Even
`flow_control`	number	✅	Hardware flow control: `1` enabled, `0` disabled

Field	Type	Writable	Description
`mode`	number	✅	Serial port working mode, see enum table

Extension Fields (present when `mode = 1` DTU mode, inferred from code)

Field	Type	Writable	Description
`protocol`	number	✅	DTU protocol type: `1`=transparent, `2`=Modbus, `3`=TCP Server, `4`=UDP Server, `5`=MQTT
`local_port`	number	✅	Local port, range 1–65535; valid when `protocol=2` (Modbus) or `protocol=3/4` (TCP/UDP Server)
`tcp_udp_protocol`	number	✅	Transparent sub-protocol: `1`=TCP, `2`=UDP; valid when `protocol=1` (transparent)
`kp_interval`	number	✅	TCP KeepAlive probe interval (seconds), range 1–3600; valid for TCP client
`kp_retries`	number	✅	TCP KeepAlive retry count, range 1–16; valid for TCP client
`fragment_length`	number	✅	Serial framing length (bytes), range 1–1024
`fragment_interval`	number	✅	Serial framing interval (ms), range 10–65535
`reconnect_interval`	number	✅	TCP reconnection interval (seconds), range 10–60; valid for TCP client
`assigned_protocol`	number	✅	Whether to use Ursalink proprietary protocol: `1` enabled, `0` disabled; valid for TCP client
`hb_interval`	number	✅	Heartbeat interval (seconds), range 10–600; valid when Ursalink protocol is enabled
`content`	string	✅	Connection identifier, up to 63 characters; valid for TCP/UDP client (used as device ID when Ursalink protocol is enabled, otherwise as registration packet content)
`server_list`	array	✅	Server address list, up to 10 entries; valid for TCP/UDP client, see sub-table

Field	Type	Writable	Description
`transpond_mqtt_list`	array	✅	MQTT forwarding rule list, up to 20 entries; valid when `protocol=5` (MQTT), see sub-table

Extension Fields (present when `mode = 1` DTU and `protocol = 2` Modbus, inferred from code)

Field	Type	Writable	Description
`max_num_tcp`	number	✅	Maximum TCP connections, range 1–128
`conn_timeout`	number	✅	Connection timeout (seconds), range 0–1000
`read_interval`	number	✅	Read interval (ms), range 1–10000
`response_time`	number	✅	Response timeout (ms), range 1–10000
`retries`	number	✅	Retry count, range 0–15

Extension Fields (`mode = 5` IEC, only for 3X/4X custom firmware, inferred from code)

Field	Type	Writable	Description
`common_address_101`	number	✅	IEC 101 ASDU address length: `1` or `2`
`common_address_104`	number	✅	IEC 104 ASDU address length: `1` or `2`
`cause_of_transmission_101`	number	✅	IEC 101 COT length: `1` or `2`
`cause_of_transmission_104`	number	✅	IEC 104 COT length: `1` or `2`
`information_object_address_101`	number	✅	IEC 101 IOA length: `1`, `2` or `3`
`information_object_address_104`	number	✅	IEC 104 IOA length: `1`, `2` or `3`
`link_mode`	number	✅	101 protocol mode: `0`=balanced mode, `1`=unbalanced mode
`link_address_size`	number	✅	101 link address length: `1` or `2`
`link_address`	number	✅	101 link address; range 1–255 when `link_address_size=1`, range 1–65535 when `=2`
`serial_debug`	number	✅	Serial debug: `1` enabled, `0` disabled

`mode` Enum Values

Value	Description	Applicable Condition
`1`	DTU (transparent)	All devices
`2`	GPS	Models with GPS module only
`3`	Modbus Client	All devices
`4`	Modbus Server	Multi-port devices (non-single-port models)
`5`	IEC	Only 3X/4X custom firmware
`6`	DLMS Connect	All devices

`serial_type` Enum Values

Value	Description
`1`	RS232
`2`	RS485

`baudrate` Enum Values

Value	Baud Rate (bps)
`1`	300
`2`	1200
`3`	2400
`4`	4800
`5`	9600
`6`	19200
`7`	38400
`8`	57600
`9`	115200
`10`	230400

`server_list[]` Sub-fields

Field	Type	Description
`ip`	string	Server IP or domain, up to 127 characters
`port`	number	Server port, range 1–65535

Field	Type	Description
`ip_status`	number	Connection status (read-only): `-2` =unknown/not applicable, `0` =not connected, `1` =connected; in UDP mode fixed as `-`

`transpond_mqtt_list[]` **Sub-fields**

Field	Type	Description
`mqtt_name`	string	MQTT connection name, selected from the system MQTT service list
`type`	number	Direction: `0` =uplink (serial→MQTT), `1` =downlink (MQTT→serial)
`topic`	string	Publish/subscribe topic, up to 511 characters
`flag`	number	Retain flag: `1` retained, `0` not retained; not applicable for downlink (`type=1`)
`qos`	number	QoS level: `0` / `1` / `2`

> A single connection (`mqtt_name` + `topic`) allows at most one uplink and one downlink rule; up to 10 uplink rules and 10 downlink rules.

4.2.1.2 Get MQTT Service Name List

Used to populate the MQTT forwarding connection name dropdown when the page loads.

Request

```
{
  "id": 17,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "get_name_list",
  "values": []
}
```

Response

```
{
  "id": 17,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_service_mqtt",
          "index": 1,
```

```
          "value": {
            "mqtt_name_list": ["123"]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

Field	Type	Description
`mqtt_name_list`	array<string>	Configured MQTT connection name list, used as connection name options for MQTT forwarding rules

---

4.2.1.3 Save Serial Port 1 Configuration

Request

The SET request `values` contains only the changed fields.

```
{
  "id": 18,
  "execute": 1,
  "core": "yruo_industrial_serial_1",
  "function": "set",
  "values": [
    {
      "base": "yruo_industrial_serial_1",
      "type": "yruo_industrial_serial_1",
      "index": 1,
      "value": {
        "enable": 1,
        "mode": 1,
        "protocol": 1,
        "tcp_udp_protocol": 1,
        "baudrate": 9,
        "server_list": [
          { "ip": "192.168.1.100", "port": 8080 }
        ]
      }
    }
  ]
}
```

The fields that can be submitted in `value` refer to all writable (✅) fields in 4.2.1.1 Field Descriptions.

> **Note**: `local_port_1` (Modbus mode) and `local_port_2` (TCP/UDP Server mode) are displayed as different labels in the UI, but both are submitted using the field name `local_port`; likewise `content1` and `content2` are both submitted as `content`.

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.2.2 Serial Port 2

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.2.2.1 Get Serial Port 2 Configuration

Request

```
{
  "id": 18,
  "execute": 1,
  "core": "yruo_industrial_serial_2",
  "function": "get",
  "values": [{ "base": "yruo_industrial_serial_2" }]
}
```

Response

The following is a complete response example for `mode=1` (DTU transparent):


```
{
  "id": 18,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_industrial_serial_2",
          "index": 1,
          "value": {
            "serial_type": 2,
            "baudrate": 5,
            "data_bits": 1,
            "stop_bits": 1,
            "parity": 1,
            "flow_control": 0,
            "mode": 1,
            "protocol": 1,
            "tcp_udp_protocol": 1,
            "kp_interval": 75,
            "kp_retries": 9,
            "fragment_length": 1024,
            "fragment_interval": 100,
            "reconnect_interval": 10,
            "assigned_protocol": 0,
            "hb_interval": 30,
            "server_list": [],
            "enable": 0
          }
        }
      ]
    }
  ]
}
```


> In non-DTU mode (`mode ≠ 1`), only basic fields are returned, without `protocol` and subsequent extension fields.

Field Descriptions

Basic Fields (always present)

Field	Type	Writable	Description
`enable`	number	✅	Whether enabled: `1` enabled, `0` disabled

Field	Type	Writable	Description
`serial_type`	number	❌	Serial port type (read-only, hardware-fixed): `1`=RS232, `2`=RS485
`baudrate`	number	✅	Baud rate, see enum table
`data_bits`	number	✅	Data bits: `1`=8bits, `2`=7bits
`stop_bits`	number	✅	Stop bits: `1`=1bit, `2`=2bits
`parity`	number	✅	Parity: `1`=None, `2`=Odd, `3`=Even
`flow_control`	number	✅	Hardware flow control: `1` enabled, `0` disabled
`mode`	number	✅	Serial port working mode, see enum table

Extension Fields (present when `mode = 1` DTU mode)

Field	Type	Writable	Description
`protocol`	number	✅	DTU protocol type: `1`=transparent, `2`=Modbus, `3`=TCP Server, `4`=UDP Server, `5`=MQTT
`local_port`	number	✅	Local port, range 1–65535; valid when `protocol=2` (Modbus) or `protocol=3/4` (TCP/UDP Server)
`tcp_udp_protocol`	number	✅	Transparent sub-protocol: `1`=TCP, `2`=UDP; valid when `protocol=1` (transparent)
`kp_interval`	number	✅	TCP KeepAlive probe interval (seconds), range 1–3600; valid for TCP client
`kp_retries`	number	✅	TCP KeepAlive retry count, range 1–16; valid for TCP client
`fragment_length`	number	✅	Serial framing length (bytes), range 1–1024
`fragment_interval`	number	✅	Serial framing interval (ms), range 10–65535
`reconnect_interval`	number	✅	TCP reconnection interval (seconds), range 10–60; valid for TCP client
`assigned_protocol`	number	✅	Whether to use Ursalink proprietary protocol: `1` enabled, `0` disabled; valid for TCP client
`hb_interval`	number	✅	Heartbeat interval (seconds), range 10–600; valid when Ursalink protocol is enabled

Field	Type	Writable	Description
`content`	string	✅	Connection identifier, up to 63 characters; valid for TCP/UDP client (used as device ID when Ursalink protocol is enabled, otherwise as registration packet content)
`server_list`	array	✅	Server address list, up to 10 entries; valid for TCP/UDP client, see sub-table
`transpond_mqtt_list`	array	✅	MQTT forwarding rule list, up to 20 entries; valid when `protocol=5` (MQTT), see sub-table

Extension Fields (present when `mode = 1` DTU and `protocol = 2` Modbus, inferred from code)

Field	Type	Writable	Description
`max_num_tcp`	number	✅	Maximum TCP connections, range 1–128
`conn_timeout`	number	✅	Connection timeout (seconds), range 0–1000
`read_interval`	number	✅	Read interval (ms), range 1–10000
`response_time`	number	✅	Response timeout (ms), range 1–10000
`retries`	number	✅	Retry count, range 0–15

`mode` Enum Values

Value	Description	Applicable Condition
`1`	DTU (transparent)	All devices
`2`	GPS	Models with GPS module only
`3`	Modbus Client	All devices
`4`	Modbus Server	Multi-port devices (non-single-port models)
`6`	DLMS Connect	All devices

> Serial Port 2 does not support IEC mode (`mode=5`), which is the main difference from Serial Port 1.

`baudrate` Enum Values

Value	Baud Rate (bps)
`1`	300
`2`	1200

Value	Baud Rate (bps)
`3`	2400
`4`	4800
`5`	9600
`6`	19200
`7`	38400
`8`	57600
`9`	115200
`10`	230400

`server_list[]` Sub-fields

Field	Type	Description
`ip`	string	Server IP or domain, up to 127 characters
`port`	number	Server port, range 1–65535
`ip_status`	number	Connection status (read-only): `-2`=unknown/not applicable, `0`=not connected, `1`=connected; in UDP mode fixed as `-`

`transpond_mqtt_list[]` Sub-fields

Field	Type	Description
`mqtt_name`	string	MQTT connection name, selected from the system MQTT service list
`type`	number	Direction: `0`=uplink (serial→MQTT), `1`=downlink (MQTT→serial)
`topic`	string	Publish/subscribe topic, up to 511 characters
`flag`	number	Retain flag: `1` retained, `0` not retained; not applicable for downlink (`type=1`)
`qos`	number	QoS level: `0` / `1` / `2`

> A single connection (`mqtt_name` + `topic`) allows at most one uplink and one downlink rule; up to 10 uplink rules and 10 downlink rules.

4.2.2.2 Get MQTT Service Name List

Used to populate the MQTT forwarding connection name dropdown when the page loads.

Request

```
{
  "id": 19,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "get_name_list",
  "values": []
}
```

Response

```
{
  "id": 19,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_service_mqtt",
          "index": 1,
          "value": {
            "mqtt_name_list": ["123"]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

Field	Type	Description
`mqtt_name_list`	array<string>	Configured MQTT connection name list, used as connection name options for MQTT forwarding rules

---

4.2.2.3 Save Serial Port 2 Configuration

Request

The SET request `values` contains only the changed fields.

```
{
  "id": 20,
  "execute": 1,
  "core": "yruo_industrial_serial_2",
  "function": "set",
  "values": [
```

```
    {
      "base": "yruo_industrial_serial_2",
      "type": "yruo_industrial_serial_2",
      "index": 1,
      "value": {
        "enable": 1,
        "mode": 1,
        "protocol": 1,
        "tcp_udp_protocol": 1,
        "baudrate": 9,
        "server_list": [
          { "ip": "192.168.1.100", "port": 8080 }
        ]
      }
    }
  ]
}
```

The fields that can be submitted in `value` refer to all writable (✅) fields in 4.2.2.1 Field Descriptions.

> **Note**: `local_port_1` (Modbus mode) and `local_port_2` (TCP/UDP Server mode) are displayed as different labels in the UI, but both are submitted using the field name `local_port`; likewise `content1` and `content2` are both submitted as `content`.

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.3 Modbus Server

4.3.1 Modbus TCP

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.3.1.1 Get Modbus TCP Server Configuration

Request

```
{
  "id": 20,
  "execute": 1,
  "core": "yruo_modbus_tcp",
  "function": "get",
  "values": [{ "base": "yruo_modbus_tcp" }]
}
```

Response

```
{
  "id": 20,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_tcp",
          "index": 1,
          "value": {
            "local_port": 502,
            "di_1_address": 0,
            "di_2_address": 1,
            "do_1_address": 0,
            "do_2_address": 1,
            "enable": 0
          }
        }
      ]
    }
```

```
  ]
}
```

Field Descriptions

Field	Type	Writable	Description
`enable`	number	✅	Whether enabled: `1` enabled, `0` disabled
`local_port`	number	✅	Listening port, range 1–65535, default `502`
`di_1_address`	number	✅	DI1 Modbus register address, range 0–255
`di_2_address`	number	✅	DI2 Modbus register address, range 0–255; not displayed on 3X/4X single-channel devices, the value must differ from `di_1_address`
`do_1_address`	number	✅	DO1 Modbus register address, range 0–255
`do_2_address`	number	✅	DO2 Modbus register address, range 0–255; not displayed on 3X/4X single-channel devices, the value must differ from `do_1_address`

4.3.1.2 Save Modbus TCP Server Configuration

Request

The SET request `values` contains only the changed fields.

```
{
  "id": 21,
  "execute": 1,
  "core": "yruo_modbus_tcp",
  "function": "set",
  "values": [
    {
      "base": "yruo_modbus_tcp",
      "type": "yruo_modbus_tcp",
      "index": 1,
      "value": {
        "enable": 1,
        "local_port": 502,
        "di_1_address": 0,
        "di_2_address": 1,
        "do_1_address": 0,
        "do_2_address": 1
      }
    }
  ]
}
```

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.3.2 Modbus RTU

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.3.2.1 Get Modbus RTU Server Configuration

Request

```
{
  "id": 21,
  "execute": 1,
  "core": "yruo_modbus_rtu",
  "function": "get",
  "values": [{ "base": "yruo_modbus_rtu" }]
}
```

Response


```
{
  "id": 21,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_rtu",
          "index": 1,
          "value": {
            "slave_id": 1,
            "serial_mode": 0,
            "serial_1_flag": 0,
            "serial_2_flag": 0,
            "di_1_address": 0,
            "di_2_address": 1,
            "do_1_address": 0,
            "do_2_address": 1,
            "enable": 0
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

Field	Type	Writable	Description
`enable`	number	✅	Whether enabled: `1` enabled, `0` disabled
`serial_mode`	number	✅	Serial port to use, see enum table
`slave_id`	number	✅	Modbus slave ID, range 1–247
`di_1_address`	number	✅	DI1 Modbus register address, range 0–255
`di_2_address`	number	✅	DI2 Modbus register address, range 0–255; not displayed on 3X/4X single-channel devices, the value must differ from `di_1_address`
`do_1_address`	number	✅	DO1 Modbus register address, range 0–255
`do_2_address`	number	✅	DO2 Modbus register address, range 0–255; not displayed on 3X single-channel devices, the value must differ from `do_1_address`

Field	Type	Writable	Description
`serial_1_flag`	number	❌	Whether Serial 1 is configured as Modbus Slave mode: `1` yes, `0` no; on save, if the flag corresponding to the selected serial port is `0`, an error is reported
`serial_2_flag`	number	❌	Whether Serial 2 is configured as Modbus Slave mode: `1` yes, `0` no; on save, if the flag corresponding to the selected serial port is `0`, an error is reported

`serial_mode` enum values:

Value	Description
`0`	Serial 1
`1`	Serial 2; only selectable on device models that support dual serial ports

---

4.3.2.2 Save Modbus RTU Server Configuration

Request

The SET request `values` contains only the changed fields.

```
{
  "id": 22,
  "execute": 1,
  "core": "yruo_modbus_rtu",
  "function": "set",
  "values": [
    {
      "base": "yruo_modbus_rtu",
      "type": "yruo_modbus_rtu",
      "index": 1,
      "value": {
        "enable": 1,
        "serial_mode": 0,
        "slave_id": 1,
        "di_1_address": 0,
        "di_2_address": 1,
        "do_1_address": 0,
        "do_2_address": 1
      }
    }
  ]
}
```

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.3.3 Modbus RTU Over TCP

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.3.3.1 Get Modbus RTU Over TCP Configuration

Request

```
{
  "id": 22,
  "execute": 1,
  "core": "yruo_modbus_rtu_over_tcp",
  "function": "get",
  "values": [{ "base": "yruo_modbus_rtu_over_tcp" }]
}
```

Response


```
{
  "id": 22,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_rtu_over_tcp",
          "index": 1,
          "value": {
            "slave_id": 1,
            "di_1_address": 0,
            "di_2_address": 1,
            "do_1_address": 0,
            "do_2_address": 1,
            "reconnect_interval": 10,
            "server_list": [],
            "enable": 0
          }
        }
      ]
    }
  ]
}
```


> The structure of `server_list` when it has entries is shown in the field descriptions below.

Field Descriptions

Field	Type	Writable	Description
`enable`	number	✅	Whether enabled: `1` enabled, `0` disabled
`slave_id`	number	✅	Modbus slave ID, range 1–247
`device_id`	string	✅	Device identifier, spaces are not allowed, up to 63 characters; not confirmed in actual response, may be a device-model-related field
`reconnect_interval`	number	✅	Reconnection interval on disconnect (seconds), range 1–3600
`di_1_address`	number	✅	DI1 Modbus register address, range 0–255

Field	Type	Writable	Description
`di_2_address`	number	✅	DI2 Modbus register address, range 0–255; not displayed on 3X/4X single-channel devices, the value must differ from `di_1_address`
`do_1_address`	number	✅	DO1 Modbus register address, range 0–255
`do_2_address`	number	✅	DO2 Modbus register address, range 0–255; not displayed on 3X/4X single-channel devices, the value must differ from `do_1_address`
`server_list`	array	✅	TCP server list, up to 10 entries, see sub-table below

`server_list[]` entry fields (GET response):

Field	Type	Description
`ip`	string	Server IP address or domain
`port`	number	Server port, range 1–65535
`ip_status`	number	Connection status: `1` connected, `0` not connected; read-only, not submittable

---

4.3.3.2 Save Modbus RTU Over TCP Configuration

Request

The SET request `values` contains only the changed fields. Entries in `server_list` require an additional `old_ip` and `old_port` to identify the original entry (the client injects these automatically before submission, with values equal to the `ip` / `port` read at the time).

```
{
  "id": 23,
  "execute": 1,
  "core": "yruo_modbus_rtu_over_tcp",
  "function": "set",
  "values": [
    {
      "base": "yruo_modbus_rtu_over_tcp",
      "type": "yruo_modbus_rtu_over_tcp",
      "index": 1,
      "value": {
        "enable": 1,
        "slave_id": 1,
        "reconnect_interval": 10,
        "di_1_address": 0,
        "di_2_address": 1,
```

```
        "do_1_address": 0,
        "do_2_address": 1,
        "server_list": [
          {
            "ip": "192.168.1.100",
            "port": 502,
            "old_ip": "192.168.1.100",
            "old_port": 502
          }
        ]
      }
    }
  ]
}
```

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.4 Modbus Client

4.4.1 Modbus Client

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version

Field	Type	Description
`status`	number	Request execution status: `0` success, `-1` error, `-2` not logged in

4.4.1.1 Get Modbus Client Configuration

Request

```
{
  "id": 23,
  "execute": 1,
  "core": "yruo_modbus_master",
  "function": "get",
  "values": [{ "base": "yruo_modbus_master" }]
}
```

Response

```
{
  "id": 23,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_master",
          "index": 1,
          "value": {
            "read_interval": 300,
            "interval": 50,
            "max_retries": 3,
            "max_response_time": 500,
            "enable": 1,
            "instruction_list": [
              "123",
              "1233",
              "3"
            ]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

Field	Type	Writable	Description
`enable`	number	✅	Whether enabled: `1` enabled, `0` disabled
`read_interval`	number	✅	Auto read interval (seconds), range 0–604800; `0` means auto read is disabled
`max_retries`	number	✅	Maximum retry count, range 0–5
`max_response_time`	number	✅	Maximum response timeout (milliseconds), range 10–1000
`interval`	number	✅	Instruction interval (milliseconds), range 10–1000
`instruction_list`	array	❌	Configured channel name list (string array), read-only; used as the data source for the channel selection dropdown on the page

---

4.4.1.2 Save Modbus Client Configuration

Request

The SET request `values` contains only the changed fields.

```
{
  "id": 24,
  "execute": 1,
  "core": "yruo_modbus_master",
  "function": "set",
  "values": [
    {
      "base": "yruo_modbus_master",
      "type": "yruo_modbus_master",
      "index": 1,
      "value": {
        "enable": 1,
        "read_interval": 300,
        "max_retries": 3,
        "max_response_time": 500,
        "interval": 50
      }
    }
  ]
}
```

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.4.1.3 Read Channel Data

Triggered by the "Read" button on the page; initiates an immediate read of a specified channel on the device.

Request

```
{
  "id": 25,
  "execute": 1,
  "core": "yruo_modbus_master",
  "function": "read",
  "values": [
    {
      "base": "yruo_modbus_master",
      "index": 1,
      "value": {
        "channel": "123"
      }
    }
  ]
}
```

Parameter	Type	Required	Description
`index`	number	Yes	Consistent with the `index` in the GET response
`channel`	string	Yes	Channel name, from an entry in `instruction_list`

Response

Success (`status: 0`):

```
{
  "status": 0,
  "result": [
```

```
    {
      "get": [
        {
          "type": "yruo_modbus_master",
          "index": 1,
          "value": {
            "result": "读取结果字符串"
          }
        }
      ]
    }
  ]
}
```

Failure (`status: -1`): `result[0].get` is empty or does not exist, and the read result is shown as empty.

Field	Type	Description
`result`	string	Immediate read result for the channel; the format is determined by the channel configuration

---

4.4.1.4 Reset Modbus Client

Triggered by the "Reset" button on the page; clears all channel settings. A confirmation dialog appears before the operation.

Request

```
{
  "id": 26,
  "execute": 1,
  "core": "yruo_modbus_channelsettings",
  "function": "del_all",
  "values": [{ "base": "yruo_modbus_channelsettings" }]
}
```

Response

Success: `result[0].templet_code` is the string `"10000000"`.

Failure: `result[0].templet_code` is another error code string.

```
{
  "status": 0,
  "result": [
    {
      "templet_code": "10000000"
    }
  ]
}
```

> Note: The success state of this interface is determined by `result[0].templet_code === "10000000"`, not the top-level `status` field.

4.4.2 Channel Setting

> The `/cgi` interface uses `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, format `multipart/form-data`.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

Channel Object Field Descriptions

The field definitions of channel entries (Channel) in each interface are as follows:

Field	Type	Required	Description
`channel_name`	string	✅	Channel name, unique identifier; UTF-8 byte length ≤ 31; allows letters, digits, Chinese characters and ASCII visible characters (excluding space, `(`, `;`)
`link_type`	number	✅	Link type, see enum table
`server_id`	number	✅	Modbus slave device address, range 1–247
`ip_address`	string	Required for TCP	Server IP address, up to 15 characters; only valid when `link_type = 3` (TCP)
`port`	number	Required for TCP	Server port, range 0–65535; only valid when `link_type = 3` (TCP)
`command_type`	number	✅	Modbus function code, see enum table

Field	Type	Required	Description
`data_type`	number	✅	Data type, see enum table
`signed`	number	✅	Signed flag: `1` signed, `0` unsigned; Float32/Float64 are fixed as signed, ASCII/HEX/BOOL are fixed as unsigned
`byte_order`	number	✅	Byte order, see enum table; `0` when not applicable
`register_address`	number	✅	Register start address, range 0–65535; start address + register count - 1 ≤ 65535
`register_count`	number	Required for read instruction	Register read count; only valid when `command_type` is a read instruction (1/2/3/4)
`register_value`	string	Required for write instruction	Write value; only valid when `command_type` is a write instruction (5/6/15/16); multiple values are separated by English commas (e.g. `"FFFF,FFFF"`)
`decimal_places`	number	No	Decimal places, range 0–9; only available for read instructions with data type INT16/INT32/INT64

`link_type` enum:

Value	Description
`1`	RS232
`2`	RS485
`3`	TCP

`command_type` enum (Modbus function code):

Value	Description	Read/Write	Available Data Types
`1`	Read Coils (01)	Read	BOOL
`2`	Read Discrete Inputs (02)	Read	BOOL
`3`	Read Holding Registers (03)	Read	INT16/INT32/INT64/Float32/Float64/ASCII/HEX
`4`	Read Input Registers (04)	Read	INT16/INT32/INT64/Float32/Float64/ASCII/HEX
`5`	Write Single Coil (05)	Write	BOOL

Value	Description	Read/Write	Available Data Types
`6`	Write Single Holding Register (06)	Write	INT16/HEX
`15`	Write Multiple Coils (15)	Write	BOOL
`16`	Write Multiple Holding Registers (16)	Write	INT16/INT32/INT64/Float32/Float64/ASCII/HEX

`data_type` enum:

Value	Description
`1`	BOOL
`2`	INT16
`3`	INT32
`4`	INT64
`5`	Float32
`6`	Float64
`7`	ASCII
`8`	HEX

`byte_order` enum (meaning varies by `data_type`):

Value	Applicable Data Type	Description
`0`	BOOL / ASCII / HEX	Not applicable (appears in actual response)
`1`	INT16	AB
`2`	INT16	BA
`1`	INT32 / Float32	AB,CD
`2`	INT32 / Float32	DC,BA
`3`	INT32 / Float32	BA,DC
`4`	INT32 / Float32	CD,AB
`1`	INT64 / Float64	AB,CD,EF,GH
`2`	INT64 / Float64	HG,FE,DC,BA
`3`	INT64 / Float64	BA,DC,FE,HG
`4`	INT64 / Float64	GH,EF,CD,AB

4.4.2.1 Get Channel List

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_channelsettings",
  "function": "get",
  "values": [
    {
      "base": "yruo_modbus_channelsettings",
      "limit": 10,
      "start": 0,
      "search": ""
    }
  ]
}
```

Parameter	Type	Required	Description
`limit`	number	Yes	Page size
`start`	number	Yes	Offset (starting from `0`)
`search`	string	No	Search keyword, matches `channel_name`

Response

```
{
  "id": 47,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_channelsettings",
          "index": 1,
          "value": {
            "instruction_list": [
              {
                "channel_name": "123",
                "server_id": 21,
                "link_type": 1,
                "command_type": 3,
                "data_type": 8,
                "byte_order": 0,
                "register_address": 65535,
                "register_count": 10,
                "register_value": "",
```

```
                "signed": 0,
                "decimal_places": 0
              },
              {
                "channel_name": "1233",
                "server_id": 123,
                "link_type": 1,
                "command_type": 4,
                "data_type": 8,
                "byte_order": 0,
                "register_address": 65510,
                "register_count": 26,
                "register_value": "",
                "signed": 0,
                "decimal_places": 0
              }
              // ... more entries, total of `total` entries
            ],
            "total": 6
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

Field	Type	Description
`instruction_list`	array	Channel object array for the current page, fields are as described in "Channel Object Field Descriptions" above
`total`	number	Total number of channels (across pages)

4.4.2.2 Add Channel

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_channelsettings",
  "function": "add",
  "values": [
    {
      "base": "yruo_modbus_channelsettings",
      "index": 1,
      "value": {
        "channel_name": "myChannel",
        "link_type": 3,
        "server_id": 1,
        "ip_address": "192.168.1.100",
        "port": 502,
```

```
        "command_type": 3,
        "data_type": 2,
        "signed": 0,
        "byte_order": 1,
        "register_address": 100,
        "register_count": 2,
        "register_value": "",
        "decimal_places": 0,
        "id": "a1b2c3d4"
      }
    }
  ]
}
```

> The `id` field is randomly generated by the frontend (an 8-character string), used to identify the entry.

Response

Success: `result[0].templet_code` is the string `"10000000"`.

Failure: `result[0].templet_code` is another error code string.

```
{
  "status": 0,
  "result": [{ "templet_code": "10000000" }]
}
```

> The success of this interface is determined by `result[0].templet_code === "10000000"`, not the top-level `status`. A maximum of 1024 channels are supported.

4.4.2.3 Update Channel

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_channelsettings",
  "function": "set",
  "values": [
    {
      "base": "yruo_modbus_channelsettings",
      "index": 1,
      "value": {
        "channel_name": "myChannel",
        "link_type": 3,
        "server_id": 1,
        "ip_address": "192.168.1.100",
        "port": 502,
        "command_type": 3,
        "data_type": 2,
        "signed": 0,
        "byte_order": 1,
```

```
        "register_address": 100,
        "register_count": 2,
        "register_value": "",
        "decimal_places": 0,
        "id": "a1b2c3d4",
        "index": 1
      }
    }
  ]
}
```

> When updating, `channel_name` is used as the unique identifier; `data_type` and `command_type` cannot be modified (grayed out in the edit dialog). The request must include the `id` (stored by the frontend) and `index` fields obtained from the GET response.

Response

Same as the add interface; on success, `result[0].templet_code === "10000000"`.

4.4.2.4 Delete Channel

Supports single and batch deletion, all using this interface.

Request

```
{
  "id": 11,
  "execute": 1,
  "core": "yruo_modbus_channelsettings",
  "function": "del",
  "values": [
    {
      "base": "yruo_modbus_channelsettings",
      "index": 1,
      "value": {
        "channel_names": ["channel1", "channel2"]
      }
    }
  ]
}
```

Parameter	Type	Description
`channel_names`	array	List of channel names to delete (string array)

Response

Same as the add interface; on success, `result[0].templet_code === "10000000"`.

4.4.2.5 Test Channel

Triggered by clicking the "Test" button in the add/edit dialog; initiates an immediate Modbus connection test for the specified channel configuration.

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_channelsettings",
  "function": "test",
  "values": [
    {
      "base": "yruo_modbus_channelsettings",
      "index": 1,
      "value": {
        "channel_name": "myChannel",
        "link_type": 3,
        "server_id": 1,
        "ip_address": "192.168.1.100",
        "port": 502,
        "command_type": 3,
        "data_type": 2,
        "signed": 0,
        "byte_order": 1,
        "register_address": 100,
        "register_count": 2,
        "channel_id": "a1b2c3d4"
      }
    }
  ]
}
```

> `channel_id` is the `id` of an existing channel; pass an empty string when testing a new channel.

Response

```
{
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_channelsettings",
          "index": 1,
          "value": {
            "status": 1,
            "result": "100",
            "error_code": "",
            "errmsg": ""
          }
        }
      ]
```

```
    }
  ]
}
```

Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Description
`status`	number	Test result: `1` success, `0` failure
`result`	string	Return value on a successful read (valid for read instructions)
`error_code`	string	Modbus error code on failure
`errmsg`	string	Error description on failure

4.4.2.6 Query Import Status

Poll this interface after file upload until `status` reaches a terminal state (`success` or `failed`).

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_channelsettings",
  "function": "import_status",
  "values": [
    {
      "base": "yruo_modbus_channelsettings",
      "task_id": "optional-task-id"
    }
  ]
}
```

Parameter	Type	Required	Description
`task_id`	string	No	Task ID returned by the upload response

Response

```
{
  "id": 47,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
```

```
          "type": "yruo_modbus_channelsettings",
          "index": 1,
          "value": {
            "status": "none",
            "module": "unknown",
            "total_count": 0,
            "success_count": 0,
            "fail_count": 0,
            "has_error_file": false
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Description
`status`	string	Import task status, see enum table
`module`	string	Module name (e.g. `"channel"`, `"unknown"`)
`total_count`	number	Total number of entries in the CSV
`success_count`	number	Number of successfully imported entries
`fail_count`	number	Number of failed entries
`has_error_file`	boolean	Whether an error log file is available for download

`status` enum:

Value	Description
`"none"`	No task
`"pending"`	Pending
`"running"`	Running
`"success"`	Import succeeded
`"failed"`	Import failed
`"unknown"`	Unknown status

4.4.2.7 Import Channel Data

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
```

Form Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	Fixed as `type=modbus_channel&file=import_csv`
`size`	File size (bytes)
`file`	Binary content of the CSV file

Response

Upload success:

```
{ "result": 1, "task_id": "<string>" }
```

Upload failure:

```
{ "result": 0, "task_id": "<string>", "reason": "<string>" }
```

> After a successful upload, you must poll the **4.4.2.6 Query Import Status** interface using `task_id` until `status` becomes `success` or `failed`.

4.4.2.8 Export / Download File

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded

sessionid=<td>&type=<type>&file=<file>
```

The three operations correspond to different `type` / `file` parameters:

Operation	`type`	`file`	Default File Name
Download import template	`modbus_channel`	`template_csv`	`modbus_channel_template.csv`
Export all channels	`modbus_channel`	`export_csv`	`modbus_channel_export.csv`
Download error log	`modbus_channel_error`	`error_csv`	`modbus_channel_errors.csv`

> The error log is only available when `has_error_file: true` (see 4.4.2.6).

Response

The response is a binary file stream:

```
Content-Disposition: attachment; filename="modbus_channel_export.csv"
Content-Type: application/octet-stream
```

> The file name is taken preferentially from `Content-Disposition: attachment; filename="..."`, falling back to the default file name in the table above.

4.4.3 Alarm Setting

> The `/cgi` interface uses `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, format `multipart/form-data`.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

Alarm Entry Field Descriptions

The field definitions of alarm entries (AlarmSetting) in each interface are as follows:

Field	Type	Description
`index`	number	Backend auto-incremented unique number
`modbus_name`	string	Read channel name, uniquely identifies the alarm
`condition`	number	Alarm trigger condition type, see enum table
`limit_max`	string	Upper threshold; valid when `condition` is GT/GE/RANGE

Field	Type	Description
`limit_min`	string	Lower threshold; valid when `condition` is LT/LE/RANGE
`limit_value`	string	Match value; valid when `condition` is EQ/NE
`alarm_action`	string	Alarm method, 2-character string, see enum table
`phone`	string / number	SMS group ID; valid when `alarm_action` includes SMS (GET returns string, SET passes number)
`email`	string / number	Email group ID; valid when `alarm_action` includes email (GET returns string, SET passes number)
`normal_content`	string	Notification content template sent when returning to normal, supports placeholders
`abnormal_content`	string	Notification content template sent when an alarm is triggered, supports placeholders
`continuous_alarm`	number	Continuous alarm: `1` enabled, `0` disabled
`write_channel`	string	Target channel name written when an alarm is triggered
`id`	string	Random string generated by the frontend (passed by frontend on add, may be present in GET response)

`condition` enum:

Value	Symbol	Description	Required Fields
`1`	`>`	Greater than (GT)	`limit_max`
`2`	`<`	Less than (LT)	`limit_min`
`3`	`~`	Range (RANGE)	`limit_min` and `limit_max`
`4`	`≥`	Greater than or equal (GE)	`limit_max`
`5`	`≤`	Less than or equal (LE)	`limit_min`
`6`	`=`	Equal (EQ)	`limit_value`
`7`	`≠`	Not equal (NE)	`limit_value`

> EQ/NE also apply to ASCII, HEX, and BOOL data types; GT/GE/LT/LE/RANGE only apply to INT16/INT32/INT64/Float32/Float64.

`alarm_action` enum (2-character string, character position 0 = SMS, character position 1 = email):

Value	Description
`"10"`	SMS only

Value	Description
`"01"`	Email only
`"11"`	SMS + email
`"00"`	No notification

Placeholders supported by `normal_content` / `abnormal_content`:

Placeholder	Description
`$YEAR`	Year
`$MON`	Month
`$DAY`	Day
`$TIME`	Time
`$VALUE`	Current collected value
`$ADDRESS`	Register address
`$NAME`	Channel name
`$CONDITION`	Alarm threshold condition description

4.4.3.1 Get Alarm List

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_alarmsettings",
  "function": "get",
  "values": [
    {
      "base": "yruo_modbus_alarmsettings",
      "limit": 10,
      "start": 0,
      "search": ""
    }
  ]
}
```

Parameter	Type	Required	Description
`limit`	number	Yes	Page size
`start`	number	Yes	Offset (starting from `0`)
`search`	string	No	Search keyword, matches `modbus_name`

Response


```
{
  "id": 47,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_alarmsettings",
          "index": 1,
          "value": {
            "alarm_list": [
              {
                "index": 1,
                "modbus_name": "123",
                "condition": 6,
                "limit_max": "",
                "limit_min": "",
                "limit_value": "1111",
                "continuous_alarm": 1,
                "phone": "3",
                "normal_content": "Note: $YEAR/$MON/$DAY $TIME, get NORMAL data
$VALUE from address $ADDRESS of channel $NAME. (Abnormal scope is  $CONDITION)",
                "abnormal_content": "Note: $YEAR/$MON/$DAY $TIME, get ABERRANT
data $VALUE from address $ADDRESS of channel $NAME. (Abnormal scope is
$CONDITION)",
                "write_channel": "22",
                "alarm_action": "10"
              }
              // ... more entries, total of `total` entries
            ],
            "total": 1,
            "phone_list": ["3"],
            "email_list": ["23"]
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Description
`alarm_list`	array	Alarm entry array for the current page, fields as described in "Alarm Entry Field Descriptions" above

Field	Type	Description
`total`	number	Total number of alarms (across pages), used for frontend pagination
`phone_list`	array	List of SMS group ID strings referenced by all current alarms
`email_list`	array	List of email group ID strings referenced by all current alarms

---

4.4.3.2 Add Alarm

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_alarmsettings",
  "function": "add",
  "values": [
    {
      "base": "yruo_modbus_alarmsettings",
      "index": 1,
      "value": {
        "id": "aB3kZ9xYqW",
        "modbus_name": "channel1",
        "condition": 6,
        "limit_max": "",
        "limit_min": "",
        "limit_value": "1111",
        "alarm_action": "10",
        "phone": 3,
        "normal_content": "Note: $YEAR/$MON/$DAY $TIME, get NORMAL data $VALUE
from address $ADDRESS of channel $NAME. (Abnormal scope is  $CONDITION)",
        "abnormal_content": "Note: $YEAR/$MON/$DAY $TIME, get ABERRANT data
$VALUE from address $ADDRESS of channel $NAME. (Abnormal scope is  $CONDITION)",
        "continuous_alarm": 1,
        "write_channel": "writeChannel1"
      }
    }
  ]
}
```

> - `id` is generated by the frontend as a 10-character random alphanumeric string.
> - `condition`, `continuous_alarm`, `phone`, and `email` are passed as number types (even if the form value is a string, the frontend performs type conversion).
> - `phone` only needs to be passed when `alarm_action` includes SMS (character position 0 is `"1"`); `email` only needs to be passed when it includes email (character position 1 is `"1"`).
> - A maximum of 1024 alarms are supported.

Response

Success: `result[0].templet_code` is the string `"10000000"`.

Failure:

```
{
  "status": 0,
  "result": [{ "templet_code": "<error_code>", "params_count": 1,
"templet_params": ["error detail"] }]
}
```

> Success is determined by `result[0].templet_code === "10000000"`, not the top-level `status`.

4.4.3.3 Update Alarm

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_alarmsettings",
  "function": "set",
  "values": [
    {
      "base": "yruo_modbus_alarmsettings",
      "index": 1,
      "value": {
        "id": "aB3kZ9xYqW",
        "index": 1,
        "modbus_name": "channel1",
        "condition": 6,
        "limit_max": "",
        "limit_min": "",
        "limit_value": "1111",
        "alarm_action": "11",
        "phone": 3,
        "email": 23,
        "normal_content": "Note: $YEAR/$MON/$DAY $TIME, get NORMAL data $VALUE
from address $ADDRESS of channel $NAME. (Abnormal scope is  $CONDITION)",
        "abnormal_content": "Note: $YEAR/$MON/$DAY $TIME, get ABERRANT data
$VALUE from address $ADDRESS of channel $NAME. (Abnormal scope is  $CONDITION)",
        "continuous_alarm": 1,
        "write_channel": "writeChannel1"
      }
    }
  ]
}
```

> `id` and `index` are taken from the corresponding entry in the GET response, used to identify the alarm to update.

Response

Same as the add interface; on success, `result[0].templet_code === "10000000"`.

4.4.3.4 Batch Delete Alarms

Supports single and batch deletion, all using this interface.

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_alarmsettings",
  "function": "del",
  "values": [
    {
      "base": "yruo_modbus_alarmsettings",
      "index": 1,
      "value": {
        "alarm_settings": [
          { "modbus_name": "channel1", "index": 1 },
          { "modbus_name": "channel2", "index": 2 }
        ]
      }
    }
  ]
}
```

Parameter	Type	Description
`alarm_settings`	array	List of alarm identifiers to delete; each item contains `modbus_name` (channel name) and `index` (number)

Response

Same as the add interface; on success, `result[0].templet_code === "10000000"`.

4.4.3.5 Query Import Status

Poll this interface after file upload until `status` reaches a terminal state (`success` or `failed`).

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_alarmsettings",
  "function": "import_status",
  "values": [
    {
      "base": "yruo_modbus_alarmsettings",
      "task_id": "optional-task-id"
    }
  ]
}
```

Parameter	Type	Required	Description
`task_id`	string	No	Task ID returned by the upload response

Response

```
{
  "id": 47,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_alarmsettings",
          "index": 1,
          "value": {
            "status": "none",
            "module": "alarm",
            "total_count": 0,
            "success_count": 0,
            "fail_count": 0,
            "has_error_file": false
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Description
`status`	string	Import task status, see enum table
`module`	string	Module name (e.g. `"alarm"`)
`total_count`	number	Total number of entries in the CSV
`success_count`	number	Number of successfully imported entries
`fail_count`	number	Number of failed entries
`has_error_file`	boolean	Whether an error log file is available for download

> If `value` is an empty object `{}`, it indicates the backend has not yet started processing the file and is treated as a failure.

`status` enum:

Value	Description
`"none"`	No task
`"pending"`	Pending
`"running"`	Running
`"success"`	Import succeeded
`"failed"`	Import failed
`"unknown"`	Unknown status

4.4.3.6 Import Alarm Data

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
```

Form Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	Fixed as `type=modbus_alarm&file=import_csv`
`size`	File size (bytes)
`file`	Binary content of the CSV file

Response

Upload success:

```
{ "result": 1, "task_id": "<string>" }
```

Upload failure:

```
{ "result": 0, "task_id": "<string>", "reason": "<string>" }
```

> After a successful upload, you must poll the **4.4.3.5 Query Import Status** interface using `task_id` until `status` becomes `success` or `failed`.

4.4.3.7 Export / Download File

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded

sessionid=<td>&type=<type>&file=<file>
```

The three operations correspond to different `type` / `file` parameters:

Operation	`type`	`file`	Default File Name
Download import template	`modbus_alarm`	`template_csv`	`modbus_alarm_template.csv`
Export all alarms	`modbus_alarm`	`export_csv`	`modbus_alarm_export.csv`
Download error log	`modbus_alarm_error`	`error_csv`	`modbus_alarm_errors.csv`

> The error log is only available when `has_error_file: true` (see 4.4.3.5).

Response

The response is a binary file stream:

```
Content-Disposition: attachment; filename="modbus_alarm_export.csv"
Content-Type: application/octet-stream
```

> The file name is taken preferentially from `Content-Disposition: attachment; filename="..."`, falling back to the default file name in the table above.

Appendix: Page Initialization Helper Interfaces

The following interfaces are called in parallel when the alarm setting page loads, used to populate form dropdown options.

A.1 Get SMS Group List

```
{
  "id": 14,
  "execute": 1,
  "core": "yruo_phone",
  "function": "get",
  "values": [{ "base": "yruo_phone" }]
}
```

The response `result[0].get[0].value.group_list` is an array of groups:

Field	Type	Description
`gid`	number	Group ID, corresponding to the `phone` field in alarm entries
`desc`	string	Group description name
`numbers`	array	List of phone numbers in the group

A.2 Get Email Group List

```
{
  "id": 16,
  "execute": 1,
  "core": "yruo_system",
  "function": "get",
  "values": [{ "base": "email" }]
}
```

The response `result[0].get[0].value.group_list` is an array of groups:

Field	Type	Description
`gid`	number	Group ID, corresponding to the `email` field in alarm entries
`desc`	string	Group description name
`emails`	array	List of email addresses in the group

4.4.4 Data Forwarding

> The `/cgi` interface uses `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, format `multipart/form-data`.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, format `application/x-www-form-urlencoded`.

This page contains two sub-modules: **MQTT Forwarding** and **TCP Forwarding**, each using an independent `core`.

> **Note**: The GET response examples are inferred from code and have not yet been verified against actual samples.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

MQTT Forwarding

4.4.4.1 Get MQTT Forwarding List

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_mqtt",
  "function": "get",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_mqtt",
      "limit": 10,
      "start": 0,
      "search": ""
    }
  ]
}
```

Parameter	Type	Required	Description
`limit`	number	Yes	Page size
`start`	number	Yes	Offset (starting from `0`)
`search`	string	No	Search keyword, matches `mqtt_name`

Response


```
{
  "id": 47,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_dataforwarding_mqtt",
          "index": 1,
          "value": {
            "mqtt_forwardings": [
              {
                "index": 1,
                "id": "aB3kZ9xYqW",
                "mqtt_name": "myMqttConn",
                "request_topic": "modbus/request",
                "response_topic": "modbus/response",
                "modbus_name": "channel1",
                "qos": 0,
                "flag": 0
              }
              // ... more entries, total of `total` entries
            ],
            "total": 1
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Description
`mqtt_forwardings`	array	MQTT forwarding entry array for the current page, fields are shown in the table below
`total`	number	Total number of entries (across pages), used for frontend pagination

MQTT forwarding entry fields:

Field	Type	Description
`index`	number	Backend auto-incremented unique number

Field	Type	Description
`id`	string	Random string generated by the frontend (10 characters)
`mqtt_name`	string	MQTT connection name (corresponds to a connection in the MQTT module)
`request_topic`	string	Request topic, optional, maximum length 511; cannot be the same as `response_topic`
`response_topic`	string	Response topic, required, maximum length 511; cannot be the same as `request_topic`
`modbus_name`	string	Read channel name; `"All Channel"` indicates all channels
`qos`	number	QoS level, see enum table
`flag`	number	Retain flag: `1` enabled, `0` disabled

`qos` enum:

Value	Description
`0`	QoS 0 (at most once)
`1`	QoS 1 (at least once)
`2`	QoS 2 (exactly once)

The topic supports the following device variable placeholders:

Placeholder	Description
`${devicename}`	Device name
`${devicesn}`	Device serial number
`${devicemac}`	Device MAC address

4.4.4.2 Add MQTT Forwarding

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_mqtt",
  "function": "add",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_mqtt",
      "index": 1,
      "value": {
        "id": "aB3kZ9xYqW",
```

```
        "mqtt_name": "myMqttConn",
        "request_topic": "modbus/request",
        "response_topic": "modbus/response",
        "modbus_name": "channel1",
        "qos": 0,
        "flag": 0
      }
    }
  ]
}
```

> - `id` is generated by the frontend as a 10-character random alphanumeric string.
> - `qos` and `flag` are passed as number types (frontend performs type conversion).
> - A maximum of 128 MQTT forwarding entries are supported.

Response

Success: `result[0].templet_code` is the string `"10000000"`.

Failure:

```
{
  "status": 0,
  "result": [{ "templet_code": "<error_code>", "params_count": 1,
"templet_params": ["error detail"] }]
}
```

> Success is determined by `result[0].templet_code === "10000000"`, not the top-level `status`.

4.4.4.3 Update MQTT Forwarding

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_mqtt",
  "function": "set",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_mqtt",
      "index": 1,
      "value": {
        "id": "aB3kZ9xYqW",
        "index": 1,
        "mqtt_name": "myMqttConn",
        "request_topic": "modbus/request",
        "response_topic": "modbus/response",
        "modbus_name": "channel1",
        "qos": 1,
        "flag": 0
      }
```

```
    }
  ]
}
```

> `id` and `index` are taken from the corresponding entry in the GET response.

Response

Same as the add interface; on success, `result[0].templet_code === "10000000"`.

4.4.4.4 Batch Delete MQTT Forwarding

Supports single and batch deletion, all using this interface.

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_mqtt",
  "function": "del",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_mqtt",
      "index": 1,
      "value": {
        "dataforwarding_mqtts": [
          { "index": 1, "modbus_name": "channel1" },
          { "index": 2, "modbus_name": "All Channel" }
        ]
      }
    }
  ]
}
```

Parameter	Type	Description
`dataforwarding_mqtts`	array	List of entries to delete; each item contains `index` (number) and `modbus_name` (channel name)

Response

Same as the add interface; on success, `result[0].templet_code === "10000000"`.

4.4.4.5 Query MQTT Forwarding Import Status

Poll this interface after file upload until `status` reaches a terminal state (`success` or `failed`).

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_mqtt",
  "function": "import_status",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_mqtt",
      "task_id": "optional-task-id"
    }
  ]
}
```

Parameter	Type	Required	Description
`task_id`	string	No	Task ID returned by the upload response

Response

The `result[0].get[0].value` field structure is the same as the alarm setting import status (4.4.3.5):

Field	Type	Description
`status`	string	Import task status: `"none"` / `"pending"` / `"running"` / `"success"` / `"failed"` / `"unknown"`
`module`	string	Module name
`total_count`	number	Total number of entries in the CSV
`success_count`	number	Number of successfully imported entries
`fail_count`	number	Number of failed entries
`has_error_file`	boolean	Whether an error log file is available for download

---

4.4.4.6 Import MQTT Forwarding Data

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
```

Form Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	Fixed as `type=modbus_mqtt&file=import_csv`
`size`	File size (bytes)

Form Field	Description
`file`	Binary content of the CSV file

Response

Upload success: `{ "result": 1, "task_id": "<string>" }`

Upload failure: `{ "result": 0, "task_id": "<string>", "reason": "<string>" }`

> After a successful upload, you must poll interface **4.4.4.5** using `task_id` until `status` becomes `success` or `failed`.

4.4.4.7 Export / Download MQTT Forwarding File

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded

sessionid=<td>&type=<type>&file=<file>
```

Operation	`type`	`file`	Default File Name
Download import template	`modbus_mqtt`	`template_csv`	`modbus_mqtt_template.csv`
Export all forwarding entries	`modbus_mqtt`	`export_csv`	`modbus_mqtt_export.csv`
Download error log	`modbus_mqtt_error`	`error_csv`	`modbus_mqtt_errors.csv`

> The error log is only available when `has_error_file: true` (see 4.4.4.5).

Response

The response is a binary file stream:

```
Content-Disposition: attachment; filename="modbus_mqtt_export.csv"
Content-Type: application/octet-stream
```

TCP Forwarding

4.4.4.8 Get TCP Forwarding List

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_tcp",
```

```
  "function": "get",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_tcp",
      "limit": 10,
      "start": 0,
      "search": ""
    }
  ]
}
```

Parameter	Type	Required	Description
`limit`	number	Yes	Page size
`start`	number	Yes	Offset (starting from `0`)
`search`	string	No	Search keyword, matches `name`

Response

```
{
  "id": 47,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_modbus_dataforwarding_tcp",
          "index": 1,
          "value": {
            "transpond_list": [
              {
                "index": 1,
                "id": "xZ9kBaYqWm",
                "name": "channel1",
                "ip": "192.168.1.100",
                "port": 5000
              }
              // ... more entries, total of `total` entries
            ],
            "total": 1
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Description
`transpond_list`	array	TCP forwarding entry array for the current page, fields are shown in the table below
`total`	number	Total number of entries (across pages), used for frontend pagination

TCP forwarding entry fields:

Field	Type	Description
`index`	number	Backend auto-incremented unique number
`id`	string	Random string generated by the frontend (10 characters)
`name`	string	Read channel name; `"All Channel"` indicates all channels
`ip`	string	Target server IPv4 address, maximum length 15 bytes
`port`	number	Target server port, range 1–65535

4.4.4.9 Add TCP Forwarding

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_tcp",
  "function": "add",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_tcp",
      "index": 1,
      "value": {
        "id": "xZ9kBaYqWm",
        "name": "channel1",
        "ip": "192.168.1.100",
        "port": 5000
      }
    }
  ]
}
```

> - `id` is generated by the frontend as a 10-character random alphanumeric string.
> - `port` is passed as a number type (frontend performs type conversion).
> - A maximum of 128 TCP forwarding entries are supported.

Response

Success: `result[0].templet_code` is the string `"10000000"`.

Failure:

```
{
  "status": 0,
  "result": [{ "templet_code": "<error_code>", "params_count": 1,
"templet_params": ["error detail"] }]
}
```

4.4.4.10 Update TCP Forwarding

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_tcp",
  "function": "set",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_tcp",
      "index": 1,
      "value": {
        "id": "xZ9kBaYqWm",
        "index": 1,
        "name": "channel1",
        "ip": "192.168.1.100",
        "port": 5000
      }
    }
  ]
}
```

> `id` and `index` are taken from the corresponding entry in the GET response.

Response

Same as the add interface; on success, `result[0].templet_code === "10000000"`.

4.4.4.11 Batch Delete TCP Forwarding

Supports single and batch deletion, all using this interface.

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_tcp",
  "function": "del",
  "values": [
```

```
    {
      "base": "yruo_modbus_dataforwarding_tcp",
      "index": 1,
      "value": {
        "dataforwarding_tcps": [
          { "index": 1, "name": "channel1" },
          { "index": 2, "name": "All Channel" }
        ]
      }
    }
  ]
}
```

Parameter	Type	Description
`dataforwarding_tcps`	array	List of entries to delete; each item contains `index` (number) and `name` (channel name)

Response

Same as the add interface; on success, `result[0].templet_code === "10000000"`.

4.4.4.12 Query TCP Forwarding Import Status

Request

```
{
  "id": 47,
  "execute": 1,
  "core": "yruo_modbus_dataforwarding_tcp",
  "function": "import_status",
  "values": [
    {
      "base": "yruo_modbus_dataforwarding_tcp",
      "task_id": "optional-task-id"
    }
  ]
}
```

Response

The field structure is the same as 4.4.4.5.

4.4.4.13 Import TCP Forwarding Data

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
```

Form Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	Fixed as `type=modbus_tcp&file=import_csv`
`size`	File size (bytes)
`file`	Binary content of the CSV file

Response

Same response format as the MQTT import interface.

4.4.4.14 Export / Download TCP Forwarding File

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded

sessionid=<td>&type=<type>&file=<file>
```

Operation	`type`	`file`	Default File Name
Download import template	`modbus_tcp`	`template_csv`	`modbus_tcp_template.csv`
Export all forwarding entries	`modbus_tcp`	`export_csv`	`modbus_tcp_export.csv`
Download error log	`modbus_tcp_error`	`error_csv`	`modbus_tcp_errors.csv`

Response

The response is a binary file stream:

```
Content-Disposition: attachment; filename="modbus_tcp_export.csv"
Content-Type: application/octet-stream
```

Appendix: Page Initialization Helper Interface

A.1 Get MQTT Connection Name List

Called when the MQTT forwarding edit dialog opens, used to populate the "MQTT Connection" dropdown options.

```
{
  "id": 39,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "get_name_list",
  "values": []
}
```

The response `result[0].get[0].value.mqtt_name_list` is a string array, where each item is an MQTT connection name.

4.5 MQTT

> The `/cgi` interface uses `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, format `multipart/form-data`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

MQTT Connection Object Field Descriptions

The field definitions of the MQTT connection object (mqtt item) in each interface are as follows:

Field	Type	Description
`mqtt_name`	string	Connection name, unique identifier, up to 63 characters
`enabled`	number	Whether enabled: `1` enabled, `0` disabled
`conn_status`	number	Current connection status, see enum table (read-only, returned by GET, ignored by SET)
`m_server_addr`	string	MQTT Broker address, up to 127 characters
`m_server_port`	number	Broker port, range 1–65535, default `1883`

Field	Type	Description
`client_id`	string	Client ID, up to 255 characters
`connect_timeout`	number	Connection timeout (seconds), range 1–65535, default `30`
`keep_alive_interval`	number	Heartbeat interval (seconds), range 1–65535, default `60`
`auto_reconnect`	number	Auto reconnect: `1` enabled, `0` disabled
`reconnect_interval`	number	Reconnection interval (seconds), range 1–65535, default `4`; only valid when `auto_reconnect=1`
`clean_session`	number	Clean session: `1` enabled, `0` disabled
`enable_user_credentials`	number	Enable user authentication: `1` enabled, `0` disabled
`username`	string	Username, up to 127 characters; only valid when `enable_user_credentials=1`
`password`	string	Password, up to 512 characters; GET returns an empty string (actual password is not returned), SET passes the new password value
`enable_tls`	number	Enable TLS: `1` enabled, `0` disabled
`mode`	number	TLS mode, see enum table; only valid when `enable_tls=1`
`used_file`	string	TLS file slot number, values `"0"`–`"9"`; used to distinguish TLS certificate files of different connections
`tls_file_exist`	number	Whether the TLS file has been uploaded: `1` uploaded, `0` not uploaded
`will_enable`	number	Enable Will Message: `1` enabled, `0` disabled
`retain_topic`	string	Will message topic, up to 511 characters; only valid when `will_enable=1`
`qos`	number	Will message QoS level; only valid when `will_enable=1`
`retain_flag`	number	Will message retain flag: `1` retained, `0` not retained; only valid when `will_enable=1`
`content`	string	Will message payload, up to 1023 characters; only valid when `will_enable=1`
`req_topic`	string	Request topic, up to 511 characters; cannot be the same as `resp_topic`

Field	Type	Description
`req_qos`	number	Request topic QoS level
`resp_topic`	string	Response topic, up to 511 characters; when set, `req_topic` cannot be empty
`resp_qos`	number	Response topic QoS level
`subscribe_flag`	number	Response topic retain flag: `1` retained, `0` not retained
`transpond_mqtt_list`	array	System topic configuration list, fields are shown in the table below

`conn_status` enum (known enum values, may be incomplete):

Value	Color	Description
`1`	Green	Connected
`2`	Red	Connection failed
`3`	Gray	Not enabled
`4`	—	Unknown (displayed as `-`)

`mode` enum:

Value	Description
`0`	Self-signed certificate (requires uploading CA certificate, client certificate, client key)
`1`	CA certificate verification (no need to upload certificate files)

`transpond_mqtt_list` entry fields:

Field	Type	Description
`id`	number	Data type number, `0`–`3` (`4` also when device has GPS)
`topic`	string	Publish topic, up to 511 characters
`trap_interval`	string	Reporting interval (seconds), range 1–65535; required when `topic` is non-empty
`flag`	number	Retain flag: `1` retained, `0` not retained
`qos`	number	QoS level: `0` / `1` / `2`

4.5.1 Get MQTT Connection List

Request

```
{
  "id": 24,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "get",
  "values": [{ "base": "yruo_service_mqtt" }]
}
```

Response

```
{
  "id": 24,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_service_mqtt",
          "index": 1,
          "value": {
            "mac": "24:e1:24:fe:b7:36",
            "mqtt": [
              {
                "enabled": 0,
                "conn_status": 3,
                "mqtt_name": "123",
                "m_server_addr": "",
                "m_server_port": 0,
                "auto_reconnect": 1,
                "clean_session": 0,
                "client_id": "",
                "connect_timeout": 30,
                "keep_alive_interval": 60,
                "reconnect_interval": 4,
                "enable_user_credentials": 0,
                "username": "",
                "password": "",
                "enable_tls": 0,
                "mode": 1,
                "used_file": "0",
                "tls_file_exist": 0,
                "will_enable": 0,
                "content": "",
                "qos": 0,
                "retain_flag": 0,
                "retain_topic": "",
```

```
                "req_qos": 0,
                "req_topic": "",
                "resp_qos": 0,
                "subscribe_flag": 0,
                "resp_topic": "",
                "transpond_mqtt_list": [
                  { "id": 0, "topic": "", "flag": 0, "trap_interval": "", "qos":
0 },
                  { "id": 1, "topic": "", "flag": 0, "trap_interval": "", "qos":
0 },
                  { "id": 2, "topic": "", "flag": 0, "trap_interval": "", "qos":
0 },
                  { "id": 3, "topic": "", "flag": 0, "trap_interval": "", "qos":
0 }
                ]
              }
              // ... more entries
            ]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Description
`mac`	string	Device MAC address, format `xx:xx:xx:xx:xx:xx`
`mqtt`	array	MQTT connection object list, fields as described in "MQTT Connection Object Field Descriptions" above

---

4.5.2 Add MQTT Connection

Request

```
{
  "id": 24,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "set",
  "values": [
    {
      "base": "yruo_service_mqtt",
      "index": 1,
      "value": {
        "mqtt": [
          {
            "action": 1,
```

```
            "mqtt_name": "myConn",
            "enabled": 1,
            "m_server_addr": "broker.example.com",
            "m_server_port": 1883,
            "client_id": "24:e1:24:fe:b7:36_ldk7z9",
            "connect_timeout": 30,
            "keep_alive_interval": 60,
            "auto_reconnect": 1,
            "reconnect_interval": 4,
            "clean_session": 0,
            "enable_user_credentials": 0,
            "username": "",
            "password": "",
            "enable_tls": 0,
            "mode": 1,
            "used_file": "0",
            "will_enable": 0,
            "retain_topic": "",
            "retain_flag": 0,
            "qos": 0,
            "content": "",
            "req_topic": "",
            "req_qos": 0,
            "resp_topic": "",
            "resp_qos": 0,
            "subscribe_flag": 0,
            "transpond_mqtt_list": [
              { "id": 0, "topic": "", "flag": 0, "trap_interval": "", "qos": 0 },
              { "id": 1, "topic": "", "flag": 0, "trap_interval": "", "qos": 0 },
              { "id": 2, "topic": "", "flag": 0, "trap_interval": "", "qos": 0 },
              { "id": 3, "topic": "", "flag": 0, "trap_interval": "", "qos": 0 }
            ]
          }
        ]
      }
    }
  ]
}
```

> - `action: 1` indicates an add operation.
> - `client_id` is automatically generated by the frontend in the format `<mac>_<timestamp_base36>`.
> - `used_file` is automatically allocated by the frontend from the available file slots (`"0"`–`"9"` not occupied by other connections).
> - The request contains the full object with all fields, not only changed fields.

Response

Success: `result[0].templet_code` is the string (or number) `10000000`.

Failure:

```
{
  "status": 0,
  "result": [{ "templet_code": "<error_code>", "params_count": 1,
"templet_params": ["error detail"] }]
}
```

4.5.3 Modify MQTT Connection

Request


```
{
  "id": 24,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "set",
  "values": [
    {
      "base": "yruo_service_mqtt",
      "index": 1,
      "value": {
        "mqtt": [
          {
            "action": 2,
            "mqtt_name": "myConn",
            "enabled": 1,
            "m_server_addr": "broker.example.com",
            "m_server_port": 1883,
            "client_id": "24:e1:24:fe:b7:36_ldk7z9",
            "connect_timeout": 30,
            "keep_alive_interval": 60,
            "auto_reconnect": 1,
            "reconnect_interval": 4,
            "clean_session": 0,
            "enable_user_credentials": 0,
            "username": "",
            "password": "",
            "enable_tls": 0,
            "mode": 1,
            "used_file": "0",
            "will_enable": 0,
            "retain_topic": "",
            "retain_flag": 0,
            "qos": 0,
            "content": "",
            "req_topic": "modbus/req",
            "req_qos": 0,
            "resp_topic": "modbus/resp",
            "resp_qos": 0,
            "subscribe_flag": 0,
            "transpond_mqtt_list": [
              { "id": 0, "topic": "sys/status", "flag": 0, "trap_interval": "60",
"qos": 0 },
              { "id": 1, "topic": "", "flag": 0, "trap_interval": "", "qos": 0 },
```

```
            { "id": 2, "topic": "", "flag": 0, "trap_interval": "", "qos": 0 },
            { "id": 3, "topic": "", "flag": 0, "trap_interval": "", "qos": 0 }
          ]
        }
      ]
    }
  ]
}
```

> - `action: 2` indicates a modify operation.
> - `mqtt_name` is the modified name; if the name has changed, an additional `mqtt_name_old` field must be included carrying the original name.

Conditional Field	Type	Description
`mqtt_name_old`	string	Original connection name; only present when `mqtt_name` is modified

Response

Same as the add interface.

4.5.4 Delete MQTT Connection

Request

```
{
  "id": 24,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "set",
  "values": [
    {
      "base": "yruo_service_mqtt",
      "index": 1,
      "value": {
        "mqtt": [
          {
            "action": 3,
            "mqtt_name": "myConn"
          }
        ]
      }
    }
  ]
}
```

> - `action: 3` indicates a delete operation.
> - Only the `action` and `mqtt_name` fields need to be passed.

Response

Same as the add interface.

4.5.5 Upload TLS Certificate File

Each MQTT connection occupies one TLS file slot via the `used_file` field (`"0"`–`"9"`). The three certificate file types of the same connection share the same `filetype` (`mqtt_tls_<fileIndex>`).

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
```

Form Field	Description
`sessionid`	Session credential, same as the `x-session-id` value
`filename`	Format `type=<filetype>&file=<file>`, parameters are shown in the table below
`size`	File size (bytes)
`file`	Binary content of the certificate file

Parameters for the three certificate file types:

Certificate Type	`filetype`	`file`	Accepted Format
CA certificate (ca.crt)	`mqtt_tls_<fileIndex>`	`CA Certificate`	`.crt`
Client certificate (client.crt)	`mqtt_tls_<fileIndex>`	`Public Key`	`.crt`
Client key (client.key)	`mqtt_tls_<fileIndex>`	`Private Key`	`.key`

> `<fileIndex>` takes the value of the `used_file` field in the MQTT connection object (`"0"`–`"9"`). After all three files are uploaded successfully, the backend sets `tls_file_exist` to `1`, with fixed file names `ca.crt`, `client.crt`, and `client.key`.

Response

Upload success:

```
{ "size": 1234, "checksum": "a1b2c3...", "filename": "ca.crt" }
```

Upload failure:

```
{ "templet_code": <number>, "params_count": <number>, "templet_params": [...] }
```

Appendix: Get MQTT Connection Name List

Used by other modules (e.g. Modbus Data Forwarding) to populate MQTT connection dropdown options.

Request

```
{
  "id": 39,
  "execute": 1,
  "core": "yruo_service_mqtt",
  "function": "get_name_list",
  "values": []
}
```

Response

`result[0].get[0].value.mqtt_name_list` is a string array, where each item is an MQTT connection name.

4.6 SNMP

4.6.1 SNMP

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.6.1.1 Get SNMP Configuration

Request

```
{
  "id": 31,
  "execute": 1,
  "core": "yruo_snmp",
```

```
  "function": "get",
  "values": [{ "base": "system" }]
}
```

Response

```
{
  "id": 31,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "system",
          "index": 0,
          "value": {
            "enable": 0,
            "port": 161,
            "snmp_version": 2,
            "location": "",
            "contact": ""
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Writable	Description
`enable`	number	✅	Whether to enable SNMP: `1` enabled, `0` disabled
`port`	number	✅	UDP listening port, range 1–65535, default `161`
`snmp_version`	number	✅	SNMP version, see enum table
`location`	string	✅	System location (sysLocation), up to 63 characters
`contact`	string	✅	System contact (sysContact), up to 63 characters

`snmp_version` enum:

Value	Description
`1`	SNMPv1
`2`	SNMPv2c
`3`	SNMPv3

4.6.1.2 Save SNMP Configuration

Request

The SET request `values` contains only the changed fields.

```
{
  "id": 31,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "set",
  "values": [
    {
      "base": "system",
      "type": "system",
      "index": 0,
      "value": {
        "enable": 1,
        "port": 161,
        "snmp_version": 2,
        "location": "Server Room",
        "contact": "admin@example.com"
      }
    }
  ]
}
```

> Note: The GET request uses `base: "system"` (instead of `base: "yruo_snmp"`); the SET request `base` and `type` are consistent with the `type` field in the GET response, both being `"system"`.

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.6.2 MIB View

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

MIB View Entry Field Descriptions

Field	Type	Description
`view_name`	string	View name, up to 63 characters
`view_type`	number	View type, see enum table
`view_oid`	string	OID, up to 63 characters (format such as `1.3.6.1.2.1.1`)

`view_type` enum:

Value	Description
`0`	included
`1`	excluded

4.6.2.1 Get MIB View List

Request

```
{
  "id": 32,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "get",
  "values": [{ "base": "view" }]
}
```

Response

Each MIB view is returned as an independent entry in `result[0].get[]`, each with `type` of `"view"` and `index` being the backend auto-incremented number.

```
{
  "id": 32,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "view",
          "index": 1,
          "value": {
            "view_name": "All",
            "view_type": 0,
            "view_oid": "1"
          }
        },
        {
          "type": "view",
          "index": 2,
          "value": {
            "view_name": "system",
            "view_type": 0,
            "view_oid": "1.3.6.1.2.1.1"
          }
        }
        // ... more entries
      ]
    }
  ]
}
```

Field Descriptions

`result[0].get[n].value` fields are described in "MIB View Entry Field Descriptions" above.

4.6.2.2 Save MIB View

> **Note**: The SET request format is inferred from code and has not yet been verified against actual samples.

After the frontend edits the table, it submits changes via this interface. Before editing, the frontend saves the original values as `old_view_name`, `old_view_type`, `old_view_oid`, and sends them together with the new values on submission, to identify the original record being modified.

Add View

```
{
  "id": 32,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "set",
  "values": [
    {
      "base": "view",
      "type": "view",
      "index": 1,
      "value": {
        "view_name": "myView",
        "view_type": 0,
        "view_oid": "1.3.6.1.2.1"
      }
    }
  ]
}
```

Modify View

When modifying, attach the `old_view_*` fields carrying the original values:

```
{
  "id": 32,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "set",
  "values": [
    {
      "base": "view",
      "type": "view",
      "index": 2,
      "value": {
        "old_view_name": "system",
        "old_view_type": 0,
        "old_view_oid": "1.3.6.1.2.1.1",
        "view_name": "system",
        "view_type": 1,
        "view_oid": "1.3.6.1.2.1.1"
```

```
      }
    }
  ]
}
```

Conditional Field	Type	Description
`old_view_name`	string	Original view name; only present when modifying
`old_view_type`	number	Original view type; only present when modifying
`old_view_oid`	string	Original OID; only present when modifying

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.6.3 VACM

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.6.3.1 Get VACM Configuration

Request

```
{
  "id": 33,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "get",
  "values": [{ "base": "vacm" }]
}
```

Response (SNMPv1/v2c mode)

When `result[0].version` is not `"v3"`, the entry type in `get[]` is `"userlist"`.

```
{
  "id": 33,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "view": ["none", "All", "system"],
      "version": "v1&v2",
      "trap_stat": {
        "enable": 0,
        "host_name": ""
      },
      "get": [
        {
          "base": "vacm",
          "index": 1,
          "type": "userlist",
          "value": {
            "community": "private",
            "permission": 1,
            "MIB_view": "All",
            "access_range": "0.0.0.0/0"
          }
        },
        {
          "base": "vacm",
          "index": 2,
          "type": "userlist",
          "value": {
            "community": "public",
            "permission": 0,
            "MIB_view": "none",
            "access_range": "0.0.0.0/0"
          }
        }
        // ... more entries
```

```
      ]
    }
  ]
}
```

Response (SNMPv3 mode)

When `result[0].version` is `"v3"`, `get[]` contains both `"v3group"` and `"v3user"` entry types, mixed together.

> **Note**: The v3 response structure is inferred from code and has not yet been verified against actual samples.

```
{
  "result": [
    {
      "view": ["none", "All", "system"],
      "version": "v3",
      "trap_stat": { "enable": 0, "host_name": "" },
      "get": [
        {
          "base": "vacm",
          "index": 1,
          "type": "v3group",
          "value": {
            "groupname": "grp1",
            "security_level": 1,
            "read_view": "All",
            "readwrite_view": "All",
            "info_view": "none"
          }
        },
        {
          "base": "vacm",
          "index": 1,
          "type": "v3user",
          "value": {
            "username": "user1",
            "user_group": "grp1",
            "auth": 1,
            "encryption": 0
          }
        }
        // ... more entries
      ]
    }
  ]
}
```

result[0] Top-level Field Descriptions

Field	Type	Description
`view`	array	List of selectable MIB view names (string array), determined by the 4.6.2 MIB View configuration

Field	Type	Description
`version`	string	Current SNMP version: `"v1&v2"` for SNMPv1/v2c, `"v3"` for SNMPv3
`trap_stat`	object	Trap configuration status, see table below; only used for UI validation, not editable on this page
`get`	array	Configuration entry list, structure varies by `version`

`trap_stat` fields:

Field	Type	Description
`enable`	number	Whether Trap is enabled: `1` enabled, `0` disabled
`host_name`	string	Trap host name (community or v3 username); when enabled, deleting the same-named entry will be rejected

SNMPv1/v2c User Entry Fields (type: "userlist")

Field	Type	Description
`community`	string	Community string, up to 63 characters, unique within the same list
`permission`	number	Permission, see enum table
`MIB_view`	string	Associated MIB view name, value taken from `result[0].view`
`access_range`	string	Allowed network access range (CIDR format), up to 18 characters, e.g. `"0.0.0.0/0"`

`permission` enum:

Value	Description
`0`	Read (read-only)
`1`	Read/Write

SNMPv3 User Group Fields (type: "v3group")

Field	Type	Description
`groupname`	string	User group name, up to 63 characters, unique
`security_level`	number	Security level, see enum table
`read_view`	string	Read view name, value taken from `result[0].view`
`readwrite_view`	string	Read-write view name, value taken from `result[0].view`

Field	Type	Description
`info_view`	string	Notification view name, value taken from `result[0].view`

`security_level` enum:

Value	Description
`0`	NoAuth (no authentication, no encryption)
`1`	AuthNoPriv (authentication, no encryption)
`2`	AuthPriv (authentication and encryption)

SNMPv3 User Fields (type: "v3user")

Field	Type	Description
`username`	string	Username, up to 63 characters, unique
`user_group`	string	Name of the user group it belongs to, value taken from the current v3group list
`auth`	number	Authentication algorithm, see enum table; fixed at `0` when `security_level=0`
`auth_password`	string	Authentication password, 8–31 characters; not returned by GET, passed on SET; only valid when `auth ≠ 0`
`encryption`	number	Encryption algorithm, see enum table; fixed at `0` when `security_level ≤ 1`
`encryption_password`	string	Encryption password, 8–31 characters; not returned by GET, passed on SET; only valid when `encryption ≠ 0`

`auth` enum:

Value	Description
`0`	None (no authentication)
`1`	SHA
`2`	MD5

`encryption` enum:

Value	Description
`0`	None (no encryption)
`1`	AES

Value	Description
`2`	DES

4.6.3.2 Save VACM Configuration

> **Note**: The SET request format is inferred from code and has not yet been verified against actual samples.

After the frontend edits the table, it submits changes via this interface. Before editing, the original identifier fields are saved as `old_*`, and sent together with the new values on submission, to identify the original record being modified.

Add/Modify SNMPv1/v2c Entry

When modifying, attach the `old_community` field (original value); when adding, do not include `old_community`:

```
{
  "id": 33,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "set",
  "values": [
    {
      "base": "vacm",
      "type": "userlist",
      "index": 1,
      "value": {
        "old_community": "private",
        "community": "private",
        "permission": 1,
        "MIB_view": "All",
        "access_range": "0.0.0.0/0"
      }
    }
  ]
}
```

Add/Modify SNMPv3 User Group

When modifying, attach `v3_groupname` (original group name); when adding, do not include `v3_groupname`:

```
{
  "id": 33,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "set",
  "values": [
    {
      "base": "vacm",
      "type": "v3group",
      "index": 1,
```

```
      "value": {
        "v3_groupname": "grp1",
        "groupname": "grp1",
        "security_level": 1,
        "read_view": "All",
        "readwrite_view": "All",
        "info_view": "none"
      }
    }
  ]
}
```

Add/Modify SNMPv3 User

When modifying, attach `v3_username` (original username):

```
{
  "id": 33,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "set",
  "values": [
    {
      "base": "vacm",
      "type": "v3user",
      "index": 1,
      "value": {
        "v3_username": "user1",
        "username": "user1",
        "user_group": "grp1",
        "auth": 1,
        "auth_password": "mypassword",
        "encryption": 0,
        "encryption_password": ""
      }
    }
  ]
}
```

Conditional Field	Description
`old_community`	Only present when modifying a userlist entry, carries the original community string
`v3_groupname`	Only present when modifying a v3group entry, carries the original user group name
`v3_username`	Only present when modifying a v3user entry, carries the original username
`auth_password`	Only valid when `auth ≠ 0`, passed on SET
`encryption_password`	Only valid when `encryption ≠ 0`, passed on SET

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.6.4 Trap

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.6.4.1 Get Trap Configuration

Request

```
{
  "id": 34,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "get",
  "values": [{ "base": "trap" }]
}
```

Response


```
{
  "id": 34,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "trap",
          "index": 0,
          "value": {
            "enable": 0,
            "snmp_version": 2,
            "snmp_newver": 2,
            "name_options": [
              {
                "version": 12,
                "available": 1,
                "name": "private"
              },
              {
                "version": 12,
                "available": 1,
                "name": "public"
              }
            ]
          }
        }
      ]
    }
  ]
}
```


> When `enable=1`, the `value` will also include `host_address`, `host_port`, `host_name`; when `snmp_version=3`, it will also include `security_level`.

Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Writable	Description
`enable`	number	✅	Whether to enable Trap: `1` enabled, `0` disabled
`snmp_version`	number	✅	SNMP version used by Trap, see enum table; must be consistent with `snmp_newver` (frontend enforces validation)

Field	Type	Writable	Description
`snmp_newver`	number	❌	Current system SNMP version (read-only metadata), determines the selectable SNMP version range
`name_options`	array	❌	List of available community/username options, see table below; for the `host_name` field selection
`host_address`	string	✅	Trap receiver address (IP or domain), up to 127 characters; may not be returned when `enable=0`
`host_port`	number	✅	Trap receiver port, range 1–65535; may not be returned when `enable=0`
`host_name`	string	✅	The community or v3 username used, taken from `name_options[].name`; may not be returned when `enable=0`
`security_level`	number	✅	Security level, see enum table; only returned and settable when `snmp_version=3`

`snmp_version` enum:

Value	Description
`1`	SNMPv1
`2`	SNMPv2c
`3`	SNMPv3

`security_level` enum (only applicable for SNMPv3):

Value	Description
`0`	NoAuth (no authentication, no encryption)
`1`	AuthNoPriv (authentication, no encryption)
`2`	AuthPriv (authentication and encryption)

`name_options` entry fields:

Field	Type	Description
`version`	number	SNMP version group: `12` for v1/v2c, `3` for v3
`available`	number	Whether available: `1` available
`name`	string	Community string (v1/v2c) or SNMPv3 username

Field	Type	Description
`option_level`	number	The maximum `security_level` allowed for the user; only present for v3 entries (not seen in samples, inferred from code)

4.6.4.2 Save Trap Configuration

Request

The SET request `values` contains only the changed fields.

```
{
  "id": 34,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "set",
  "values": [
    {
      "base": "trap",
      "type": "trap",
      "index": 0,
      "value": {
        "enable": 1,
        "snmp_version": 2,
        "host_address": "192.168.1.100",
        "host_port": 162,
        "host_name": "public"
      }
    }
  ]
}
```

When `snmp_version=3` and `security_level` changes, the frontend `beforeSave` hook automatically attaches the following fields:

Condition	Additional Fields
`security_level=1` (AuthNoPriv)	`auth_type: 1`, `auth_pw: "password"`
`security_level=2` (AuthPriv)	`auth_type: 1`, `auth_pw: "password"`, `encry_type: 1`, `encry_pw: "password"`

> The above `auth_pw` / `encry_pw` are placeholder values hardcoded by the frontend, processed by the backend as references to the existing authentication passwords of the corresponding v3user.

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.6.5 MIB

> The `/cgi` interface uses `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.6.5.1 Get MIB File List

Request

```
{
  "id": 7,
  "execute": 1,
  "core": "yruo_snmp",
  "function": "get",
  "values": [{ "base": "mib" }]
}
```

Response


```
{
  "id": 7,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "mib",
          "index": 0,
          "value": {
            "files": [
              "LTE-ROUTER-MIB.txt"
            ]
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

Field	Type	Description
`files`	array	MIB file name list (string array), each item is a downloadable MIB file name

---

4.6.5.2 Download MIB File

Request

```
POST /cgi-bin/file-export HTTP/1.1
Content-Type: application/x-www-form-urlencoded

sessionid=<td>&type=mib&file=LTE-ROUTER-MIB.txt
```

Field	Description
`sessionid`	Session credential, same as the request header `x-session-id` value
`type`	Fixed as `mib`
`file`	File name, taken from `result[0].get[0].value.files[]`

Response

The response is a binary file stream:

```
Content-Disposition: attachment; filename="LTE-ROUTER-MIB.txt"
Content-Type: application/octet-stream
```

The file name is taken preferentially from `Content-Disposition: attachment; filename="..."`, falling back to the `file` field value in the request.

4.7 TR-069

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product Number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

4.7.1 Get TR-069 Configuration

Request

```
{
  "id": 29,
  "execute": 1,
  "core": "yruo_tr069",
  "function": "get",
  "values": [{ "base": "yruo_tr069" }]
}
```

Response

```
{
  "id": 29,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
```

```
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_tr069",
          "index": 0,
          "value": {
            "enable": 0,
            "inform_time": "-",
            "url": "",
            "acs_username": "",
            "acs_password": "",
            "inform_enable": 1,
            "inform_interval": 300,
            "cpe_username": "",
            "cpe_password": ""
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`result[0].get[0].value` fields:

Field	Type	Writable	Description
`enable`	number	✅	Whether to enable TR-069: `1` enabled, `0` disabled
`inform_time`	string	❌	Last Inform timestamp, format `YYYY-MM-DD HH:mm:ss`; `"-"` when not connected
`url`	string	✅	ACS server URL, up to 256 characters, required
`acs_username`	string	✅	ACS username, up to 256 characters
`acs_password`	string	✅	ACS password, up to 256 characters; GET returns an empty string (actual password is not returned), SET passes the new password value
`inform_enable`	number	✅	Enable periodic Inform: `1` enabled, `0` disabled
`inform_interval`	number	✅	Inform reporting interval (seconds), range 5–86400, default `300`
`cpe_username`	string	✅	CPE username, up to 256 characters

Field	Type	Writable	Description
`cpe_password`	string	✅	CPE password, up to 256 characters; GET returns an empty string (actual password is not returned), SET passes the new password value

---

4.7.2 Save TR-069 Configuration

Request

The SET request `values` contains only the changed fields.

```
{
  "id": 29,
  "execute": 1,
  "core": "yruo_tr069",
  "function": "set",
  "values": [
    {
      "base": "yruo_tr069",
      "type": "yruo_tr069",
      "index": 0,
      "value": {
        "enable": 1,
        "url": "http://acs.example.com:7547/",
        "acs_username": "admin",
        "acs_password": "newpassword",
        "inform_enable": 1,
        "inform_interval": 300,
        "cpe_username": "cpeuser",
        "cpe_password": "cpepassword"
      }
    }
  ]
}
```

Response

Success: The top-level `status` is `0`, and `result[0].get` is an empty array or does not exist.

Failure:

```
{
  "status": 0,
  "result": [
    {
      "templet_code": 1,
      "params_count": 1,
      "templet_params": ["error detail"]
    }
  ]
}
```

4.8 DLMS

4.8.1 Physical Device Setting

URL: `POST https://<device-ip>/cgi`
Header: `x-session-id: <session-id>`
Core: `yruo_dlms`
Base: `dlms_server`

Get Device List

Request

```
{
  "id": "yruo_dlms",
  "function": "get",
  "base": "dlms_server",
  "param": {
    "limit": 10,
    "start": 0
  }
}
```

Field	Type	Description
`limit`	integer	Page size
`start`	integer	Starting offset

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "base": "dlms_server",
          "value": {
            "server_count": 0,
            "server": []
          }
        }
      ]
    }
  ]
```

```
}
```

Field	Type	Description
`server_count`	integer	Total number of physical devices
`server`	array	Physical device list, see table below

server Object Fields

Field	Type	Description
`server_id`	integer	Device identifier, used for edit/delete/test
`serial_name`	string	Serial port name
`enable`	integer	Whether enabled, `0`=disabled, `1`=enabled
`device_name`	string	Device name, up to 127 characters
`server_addr`	integer	Server address, range 0–255
`lg_server_addr`	integer	Logical server address, range 0–255
`client_addr`	integer	Client address, range 0–255
`access_security`	integer	Access security level: `0`=None, `1`=Low Level, `5`=GMAC (High Level)
`password`	string	Access password (used with Low Level)
`transport_security`	integer	Transport security level: `0`=None, `2`=Encryption, `3`=Authentication + Encryption
`ic_obis`	string	Interface class OBIS code
`authentication_key`	string	Authentication key, up to 32 characters
`block_cipher_key`	string	Block cipher key, up to 32 characters
`dedicated_key`	string	Dedicated key, up to 32 characters
`lg_name`	string	Logical name
`status`	integer	**Read-only**, connection status: `3`=Connected, `1` / `2`=Connecting, `0`=Disabled, `-1`=Error
`connect_time`	string	**Read-only**, connection time

Add / Edit Device

Request

```
{
  "id": "yruo_dlms",
  "function": "set",
  "base": "dlms_server",
  "param": {
    "server_id": 1,
    "serial_name": "serial1",
    "enable": 1,
    "device_name": "Device 1",
    "server_addr": 1,
    "lg_server_addr": 1,
    "client_addr": 16,
    "access_security": 0,
    "password": "",
    "transport_security": 0,
    "ic_obis": "0.0.40.0.0.255",
    "authentication_key": "",
    "block_cipher_key": "",
    "dedicated_key": "",
    "lg_name": ""
  }
}
```

> When adding, do not pass `server_id`; when editing, `server_id` is required.

Field	Type	Required	Description
`server_id`	integer	Required for edit	Device identifier
`serial_name`	string	Yes	Serial port name
`enable`	integer	Yes	`0`=disabled, `1`=enabled
`device_name`	string	Yes	Device name, up to 127 characters
`server_addr`	integer	Yes	Server address, 0–255
`lg_server_addr`	integer	Yes	Logical server address, 0–255
`client_addr`	integer	Yes	Client address, 0–255
`access_security`	integer	Yes	`0`=None, `1`=Low Level, `5`=GMAC
`password`	string	No	Low Level security password
`transport_security`	integer	Yes	`0`=None, `2`=Encryption, `3`=Auth+Encryption
`ic_obis`	string	No	Interface class OBIS code

Field	Type	Required	Description
`authentication_key`	string	No	Authentication key, up to 32 characters
`block_cipher_key`	string	No	Block cipher key, up to 32 characters
`dedicated_key`	string	No	Dedicated key, up to 32 characters
`lg_name`	string	No	Logical name

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": []
}
```

Delete Device

Request

```
{
  "id": "yruo_dlms",
  "function": "del",
  "base": "dlms_server",
  "param": {
    "server_id": 1
  }
}
```

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": []
}
```

Test Connection

Request

```
{
  "id": "yruo_dlms",
  "function": "test",
  "base": "dlms_server",
  "param": {
    "server_id": 1
  }
}
```

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": []
}
```

4.8.2 COSEM Group Setting

URL: `POST https://<device-ip>/cgi`
Header: `x-session-id: <session-id>`
Core: `yruo_dlms`
Base: `dlms_cosem`

Get COSEM Group List

Request

```
{
  "id": "yruo_dlms",
  "function": "get",
  "base": "dlms_cosem",
  "param": {}
}
```

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
```

```
  "result": [
    {
      "get": [
        {
          "base": "dlms_cosem",
          "value": {
            "cosem_count": 0,
            "cosem": []
          }
        }
      ]
    }
  ]
}
```

Field	Type	Description
`cosem_count`	integer	Total number of COSEM groups, up to 10 groups
`cosem`	array	COSEM group list, see table below

cosem Object Fields

Field	Type	Description
`cosem_id`	integer	Group identifier, used for edit/delete/test
`enable`	integer	Whether enabled, `0`=disabled, `1`=enabled
`cosem_name`	string	Group name, up to 127 characters
`interval`	integer	Collection interval (seconds), range 1–31622400
`objects`	array	COSEM object list, up to 20 per group, see table below

objects Object Fields

Field	Type	Description
`enable`	integer	Whether enabled, `0`=disabled, `1`=enabled
`name`	string	OBIS data name, up to 127 characters
`device_id`	array	Array of associated physical device IDs
`class_id`	integer	COSEM interface class ID
`obis_code`	string	OBIS code (`A.B.C.D.E.F` format when referenced by logical name, otherwise a value of 0–65535)
`short_name`	string	Short name
`enteries`	integer	Number of entries, range 1–32767 (only takes effect when `class_id` = 7)

Add / Edit COSEM Group

Request

```
{
  "id": "yruo_dlms",
  "function": "set",
  "base": "dlms_cosem",
  "param": {
    "cosem_id": 1,
    "enable": 1,
    "cosem_name": "Group1",
    "interval": 60,
    "objects": [
      {
        "enable": 1,
        "name": "ActiveEnergyImport",
        "device_id": [1],
        "class_id": 3,
        "obis_code": "1.0.1.8.0.255",
        "short_name": "",
        "enteries": 1
      }
    ]
  }
}
```

> When adding, do not pass `cosem_id`; when editing, `cosem_id` is required.

Field	Type	Required	Description
`cosem_id`	integer	Required for edit	Group identifier
`enable`	integer	Yes	`0`=disabled, `1`=enabled
`cosem_name`	string	Yes	Group name, up to 127 characters
`interval`	integer	Yes	Collection interval (seconds), 1–31622400
`objects`	array	Yes	COSEM object list, up to 20

objects Fields

Field	Type	Required	Description
`enable`	integer	Yes	`0`=disabled, `1`=enabled
`name`	string	Yes	OBIS data name, up to 127 characters
`device_id`	array	Yes	Physical device ID array
`class_id`	integer	Yes	COSEM interface class ID
`obis_code`	string	Yes	OBIS code
`short_name`	string	No	Short name

Field	Type	Required	Description
`enteries`	integer	Conditionally required	Number of entries, 1–32767 (required when `class_id`=7)

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": []
}
```

Delete COSEM Group

Request

```
{
  "id": "yruo_dlms",
  "function": "del",
  "base": "dlms_cosem",
  "param": {
    "cosem_id": 1
  }
}
```

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": []
}
```

Test COSEM Group

Request

```
{
  "id": "yruo_dlms",
  "function": "test",
  "base": "dlms_cosem",
  "param": {
    "cosem_id": 1
```

```
  }
}
```

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": []
}
```

OBIS Scan

Scans the OBIS objects available on the specified physical device.

Base: `dlms_scan`

Request

```
{
  "id": "yruo_dlms",
  "function": "get",
  "base": "dlms_scan",
  "param": {
    "limit": 10,
    "start": 0,
    "search": "",
    "device_id": [1]
  }
}
```

Field	Type	Description
`limit`	integer	Page size
`start`	integer	Starting offset
`search`	string	Search keyword (optional)
`device_id`	array	Array of physical device IDs to scan

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
```

```
  "result": [
    {
      "get": [
        {
          "base": "dlms_scan",
          "value": {
            "obis_count": 2,
            "obis_arr": [
              {
                "obis_code": "1.0.1.8.0.255",
                "short_name": "",
                "class_id": 3
              },
              {
                "obis_code": "1.0.2.8.0.255",
                "short_name": "",
                "class_id": 3
              }
            ]
          }
        }
      ]
    }
  ]
}
```


Field	Type	Description
`obis_count`	integer	Total number of OBIS objects scanned
`obis_arr`	array	OBIS object list
`obis_arr[].obis_code`	string	OBIS code
`obis_arr[].short_name`	string	Short name
`obis_arr[].class_id`	integer	COSEM interface class ID

4.8.3 Platform Connection Setting

URL: `POST https://<device-ip>/cgi`
Header: `x-session-id: <session-id>`
Core: `yruo_dlms`
Base: `dlms_forward`

Supports up to 3 forwarding configurations.

Get Forwarding Configuration List

Request

```
{
  "id": "yruo_dlms",
  "function": "get",
  "base": "dlms_forward",
  "param": {}
}
```

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "base": "dlms_forward",
          "value": {
            "forward_count": 1,
            "forward": [
              {
                "forward_id": 1,
                "enable": 1,
                "name": "Forward1",
                "type": 0,
                "status": 0,
                "http": {
                  "server_addr": "http://192.168.1.100/api",
                  "retry": 1,
                  "retry_count": 3,
                  "timeout": 5,
                  "http_headers": []
                },
                "mqtt": {
                  "mqtt_name": "",
                  "topic": "",
                  "retain": 0,
                  "qos": 0
                },
                "msg_config": {
                  "format_type": 0,
                  "data_filter": 0,
                  "json_flags": 127,
                  "send_as_object": 0,
                  "format_string": "",
                  "cosem_groups": [],
                  "invert": 0
```

```
                }
              }
            ]
          }
        }
      ]
    }
  ]
}
```

Field	Type	Description
`forward_count`	integer	Total number of forwarding configurations, up to 3
`forward`	array	Forwarding configuration list, see table below

forward Object Fields

Field	Type	Description
`forward_id`	integer	Configuration identifier, used for edit/delete
`enable`	integer	Whether enabled, `0`=disabled, `1`=enabled
`name`	string	Configuration name, up to 127 characters
`type`	integer	Forwarding type, `0`=HTTP, `1`=MQTT
`status`	integer	**Read-only**, connection status: `1`=Connected, `0`=Disabled, `-1`=Error
`http`	object	HTTP configuration (effective when `type`=0), see table below
`mqtt`	object	MQTT configuration (effective when `type`=1), see table below
`msg_config`	object	Message configuration, see table below

http Object Fields

Field	Type	Description
`server_addr`	string	Server address (URL), up to 127 characters
`retry`	integer	Whether to enable retry, `0`=no, `1`=yes
`retry_count`	integer	Number of retries, range 1–10
`timeout`	integer	Timeout (seconds), range 1–10
`http_headers`	array	Custom HTTP request header array (strings), up to 30 entries

mqtt Object Fields

Field	Type	Description
`mqtt_name`	string	MQTT configuration name
`topic`	string	Publish topic, up to 511 characters
`retain`	integer	Whether to retain the message, `0`=no, `1`=yes
`qos`	integer	QoS level, `0` / `1` / `2`

msg_config Object Fields

Field	Type	Description
`format_type`	integer	Message format, `0`=JSON, `1`=Custom
`data_filter`	integer	Data filtering, `0`=All, `1`=Custom
`json_flags`	integer	JSON field bitmask (0–127), see table below
`send_as_object`	integer	Whether to send as object, `0`=no, `1`=yes
`format_string`	string	Custom format string (effective when `format_type`=1), up to 1023 characters
`cosem_groups`	array	Array of COSEM group IDs for custom filtering (effective when `data_filter`=1)
`invert`	integer	Whether to invert the filter, `0`=no, `1`=yes

json_flags Bitmask Description

Bit	Value	Meaning
bit0	1	COSEM Group Name
bit1	2	Timestamp
bit2	4	Date (dd/mm/yyyy)
bit3	8	Date (ISO 8601)
bit4	16	Serial Number
bit5	32	COSEM Data Size
bit6	64	COSEM Data

> `json_flags`=127 means all fields are enabled (1+2+4+8+16+32+64=127).

Add / Edit Forwarding Configuration

Request (HTTP type)

```
{
  "id": "yruo_dlms",
  "function": "set",
  "base": "dlms_forward",
  "param": {
    "forward_id": 1,
    "enable": 1,
    "name": "Forward1",
    "type": 0,
    "http": {
      "server_addr": "http://192.168.1.100/api",
      "retry": 1,
      "retry_count": 3,
      "timeout": 5,
      "http_headers": ["Authorization: Bearer token123"]
    },
    "msg_config": {
      "format_type": 0,
      "data_filter": 0,
      "json_flags": 127,
      "send_as_object": 0,
      "format_string": "",
      "cosem_groups": [],
      "invert": 0
    }
  }
}
```

Request (MQTT type)

```
{
  "id": "yruo_dlms",
  "function": "set",
  "base": "dlms_forward",
  "param": {
    "forward_id": 2,
    "enable": 1,
    "name": "Forward2",
    "type": 1,
    "mqtt": {
      "mqtt_name": "MyMQTT",
      "topic": "dlms/data",
      "retain": 0,
      "qos": 1
    },
    "msg_config": {
      "format_type": 0,
      "data_filter": 1,
      "json_flags": 67,
      "send_as_object": 1,
      "format_string": "",
```

```
      "cosem_groups": [1, 2],
      "invert": 0
    }
  }
}
```

> When adding, do not pass `forward_id`; when editing, `forward_id` is required.
> Fill in the `http` sub-object when `type`=0, and the `mqtt` sub-object when `type`=1.

Field	Type	Required	Description
`forward_id`	integer	Required for edit	Configuration identifier
`enable`	integer	Yes	`0`=disabled, `1`=enabled
`name`	string	Yes	Configuration name, up to 127 characters
`type`	integer	Yes	`0`=HTTP, `1`=MQTT
`http`	object	Required when `type`=0	HTTP configuration
`mqtt`	object	Required when `type`=1	MQTT configuration
`msg_config`	object	Yes	Message configuration

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": []
}
```

Delete Forwarding Configuration

Request

```
{
  "id": "yruo_dlms",
  "function": "del",
  "base": "dlms_forward",
  "param": {
    "forward_id": 1
  }
}
```

Response

```
{
  "id": "yruo_dlms",
  "model": "R3000 Lite",
  "pn": "...",
  "oem": "...",
  "rtver": "3.0.0",
  "status": 0,
  "result": []
}
```

4.9 GPS

4.9.1 GPS Setting

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product Number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

---

4.9.1.1 Query GPS Settings

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_industrial_gps",
  "values": [
    { "base": "yruo_industrial_gps" }
  ]
}
```

Response


```
{
  "id": 23,
  "model": "UR41",
  "pn": "L08EU0011111DE0000000000",
  "oem": "0000",
  "rtver": "41.0.0.7-a2",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_industrial_gps",
          "index": 1,
          "value": {
            "enable": 0,
            "gps_serial_enable": 0,
            "gps_network_enanle": 0,
            "gps_mqtt_enable": 0,
            "gps_debug": 0
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`yruo_industrial_gps` **type (type=** `yruo_industrial_gps`**, index=** `1`**)**

Field	Type	Description
`enable`	integer	GPS global enable, `0`=disabled, `1`=enabled; cannot be disabled when any of serial forwarding, IP forwarding, or MQTT forwarding is enabled
`gps_serial_enable`	integer	Whether serial forwarding is enabled (read-only), `0`=no, `1`=yes
`gps_network_enanle`	integer	Whether IP forwarding is enabled (read-only), `0`=no, `1`=yes; note: the field name has a spelling error in the original (`enanle`)
`gps_mqtt_enable`	integer	Whether MQTT forwarding is enabled (read-only), `0`=no, `1`=yes
`gps_debug`	integer	Debug mode, `0`=disabled, `1`=enabled; forced disabled when `enable`=0

4.9.1.2 Save GPS Settings

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_industrial_gps",
  "values": [
    {
      "base": "yruo_industrial_gps",
      "type": "yruo_industrial_gps",
      "index": 1,
      "value": {
        "enable": 1
      }
    }
  ]
}
```

The `value` object only needs to contain the changed fields. `gps_serial_enable`, `gps_network_enanle`, and `gps_mqtt_enable` are read-only status fields and cannot be submitted.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

4.9.2 IP Forwarding

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product Number

Field	Type	Description
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0`=success, `-1`=failure, `-2`=not logged in

---

4.9.2.1 Query GPS IP Forwarding Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_industrial_gps_network",
  "values": [
    { "base": "yruo_industrial_gps_network" }
  ]
}
```

Response

```
{
  "id": 24,
  "model": "UR41",
  "pn": "L08EU0011111DE0000000000",
  "oem": "0000",
  "rtver": "41.0.0.7-a2",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_industrial_gps_network",
          "index": 1,
          "value": {
            "gps_enable": 0,
            "type": "client",
            "protocol": "tcp",
            "hb_interval": 75,
            "hb_retries": 9,
            "reconnect_interval": 10,
            "server_list": [],
            "trap_interval": 30,
            "RMC": 1,
            "GSA": 1,
            "GGA": 1,
            "GSV": 1,
            "enable": 0
          }
        }
      ]
    }
```

```
  ]
}
```

Field Descriptions

`yruo_industrial_gps_network` **type (type=**`yruo_industrial_gps_network`**, index=**`1`**)**

Field	Type	Description
`gps_enable`	integer	Whether GPS is globally enabled (read-only), `0`=no, `1`=yes; enabling IP forwarding requires GPS to be enabled first
`enable`	integer	IP forwarding enable, `0`=disabled, `1`=enabled
`type`	string	Connection mode, `"client"`=client, `"server"`=server
`protocol`	string	Transport protocol, `"tcp"` / `"udp"`; only valid when `type`=`"client"`
`hb_interval`	integer	Heartbeat interval (seconds), range 1–3600; valid when `type`=`"client"` and `protocol`=`"tcp"`, or when `type`=`"server"`
`hb_retries`	integer	Heartbeat retry count, range 1–16; same as above
`local_port`	integer	Local listening port, range 1–65535; valid when `type`=`"server"` (not present in actual response, inferred from code)
`reconnect_interval`	integer	Reconnection interval (seconds), range 10–60; valid when `type`=`"client"` and `protocol`=`"tcp"`
`trap_interval`	integer	NMEA reporting interval (seconds), range 1–65535
`RMC`	integer	Whether to forward NMEA RMC sentence, `0`=no, `1`=yes
`GSA`	integer	Whether to forward NMEA GSA sentence, `0`=no, `1`=yes
`GGA`	integer	Whether to forward NMEA GGA sentence, `0`=no, `1`=yes
`GSV`	integer	Whether to forward NMEA GSV sentence, `0`=no, `1`=yes; at least one of RMC/GSA/GGA/GSV must be checked
`prefix_content`	string	NMEA data prefix, up to 63 characters (not present in actual response, inferred from code)
`suffix_content`	string	NMEA data suffix, up to 63 characters (not present in actual response, inferred from code)
`server_list`	array	Target server list, up to 5 entries; valid when `type`=`"client"`, see table below

`server_list` **entry fields**

Field	Type	Description
`ip`	string	Server IP or domain, up to 127 characters
`port`	integer	Server port, range 1–65535
`ip_status`	integer	Connection status, see enum table; only meaningful when `protocol` = `"tcp"`

`ip_status` **Enum Values**

Value	Meaning
`-2`	Unknown / not applicable in UDP mode
`0`	Not connected
`1`	Connected

4.9.2.2 Save GPS IP Forwarding Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_industrial_gps_network",
  "values": [
    {
      "base": "yruo_industrial_gps_network",
      "type": "yruo_industrial_gps_network",
      "index": 1,
      "value": {
        "enable": 1,
        "type": "client",
        "protocol": "tcp",
        "hb_interval": 75,
        "hb_retries": 9,
        "reconnect_interval": 10,
        "trap_interval": 30,
        "RMC": 1,
        "GGA": 1,
        "GSA": 0,
        "GSV": 0,
        "server_list": [
          { "ip": "192.168.1.100", "port": 5000 }
        ]
      }
    }
  ]
}
```

`value` only needs to contain the changed fields. `gps_enable` is a read-only status field and cannot be submitted.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

4.9.3 Serial Forwarding

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product Number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

---

4.9.3.1 Query GPS Serial Forwarding Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_industrial_gps_serial",
  "values": [
    { "base": "yruo_industrial_gps_serial" }
  ]
}
```

Response


```
{
  "id": 25,
  "model": "UR41",
  "pn": "L08EU0011111DE0000000000",
  "oem": "0000",
  "rtver": "41.0.0.7-a2",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_industrial_gps_serial",
          "index": 1,
          "value": {
            "gps_enable": 0,
            "serial": 1,
            "trap_interval": 30,
            "RMC": 1,
            "GSA": 1,
            "GGA": 1,
            "GSV": 1,
            "enable": 0
          }
        }
      ]
    }
  ]
}
```


Field Descriptions

`yruo_industrial_gps_serial` **type (type=**`yruo_industrial_gps_serial`**, index=**`1`**)**

Field	Type	Description
`gps_enable`	integer	Whether GPS is globally enabled (read-only), `0`=no, `1`=yes; enabling serial forwarding requires GPS to be enabled first
`enable`	integer	Serial forwarding enable, `0`=disabled, `1`=enabled
`serial`	integer	Target serial port, see enum table
`trap_interval`	integer	NMEA reporting interval (seconds), range 1–60
`RMC`	integer	Whether to forward NMEA RMC sentence, `0`=no, `1`=yes
`GSA`	integer	Whether to forward NMEA GSA sentence, `0`=no, `1`=yes
`GGA`	integer	Whether to forward NMEA GGA sentence, `0`=no, `1`=yes
`GSV`	integer	Whether to forward NMEA GSV sentence, `0`=no, `1`=yes; at least one of RMC/GSA/GGA/GSV must be checked

`serial` Enum Values

Value	Meaning
`1`	Serial 1 (displayed as "Serial" on single-serial-port devices)
`2`	Serial 2 (only supported on dual-serial-port devices)

4.9.3.2 Save GPS Serial Forwarding Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_industrial_gps_serial",
  "values": [
    {
      "base": "yruo_industrial_gps_serial",
      "type": "yruo_industrial_gps_serial",
      "index": 1,
      "value": {
        "enable": 1,
        "serial": 1,
        "trap_interval": 30,
        "RMC": 1,
        "GSA": 1,
        "GGA": 1,
        "GSV": 0
      }
    }
  ]
}
```

`value` only needs to contain the changed fields. `gps_enable` is a read-only status field and cannot be submitted.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

4.9.4 MQTT Forwarding

> All requests use `POST https://<device-ip>/cgi` uniformly, carrying the session credential via the request header `x-session-id: <td>`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	integer	Request ID
`model`	string	Device model
`pn`	string	Product Number
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	integer	Status code, `0` =success, `-1` =failure, `-2` =not logged in

4.9.4.1 Query MQTT Connection Name List

When the page loads, the MQTT connection name list is fetched first, used to populate the dropdown options in the forwarding entries.

Request

```
{
  "id": 1,
  "function": "get_name_list",
  "core": "yruo_service_mqtt",
  "values": []
}
```

Response

```
{
  "status": 0,
  "result": [
    {
      "get": [
        {
          "value": {
            "mqtt_name_list": ["mqtt-conn-1", "mqtt-conn-2"]
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

Field	Type	Description
`result[0].get[0].value.mqtt_name_list`	array<string>	List of available MQTT connection names, used as dropdown options for `transpond_mqtt_list[].mqtt_name`

4.9.4.2 Query GPS MQTT Forwarding Configuration

Request

```
{
  "id": 1,
  "function": "get",
  "core": "yruo_industrial_gps_mqtt",
  "values": [
    { "base": "yruo_industrial_gps_mqtt" }
  ]
}
```

Response

```
{
  "id": 26,
  "model": "UR41",
  "pn": "L08EU0011111DE0000000000",
  "oem": "0000",
  "rtver": "41.0.0.7-a2",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "yruo_industrial_gps_mqtt",
          "index": 1,
          "value": {
            "gps_enable": 0,
            "trap_interval": 30,
            "RMC": 1,
            "GSA": 1,
            "GGA": 1,
            "GSV": 1,
            "enable": 0,
            "transpond_mqtt_list": []
          }
        }
      ]
    }
  ]
}
```

Field Descriptions

`yruo_industrial_gps_mqtt` **type (type=** `yruo_industrial_gps_mqtt`**, index=** `1`**)**

Field	Type	Description
`gps_enable`	integer	Whether GPS is globally enabled (read-only), `0`=no, `1`=yes; enabling MQTT forwarding requires GPS to be enabled first
`enable`	integer	MQTT forwarding enable, `0`=disabled, `1`=enabled
`trap_interval`	integer	NMEA reporting interval (seconds), range 1–65535
`RMC`	integer	Whether to forward NMEA RMC sentence, `0`=no, `1`=yes
`GSA`	integer	Whether to forward NMEA GSA sentence, `0`=no, `1`=yes
`GGA`	integer	Whether to forward NMEA GGA sentence, `0`=no, `1`=yes
`GSV`	integer	Whether to forward NMEA GSV sentence, `0`=no, `1`=yes; at least one of RMC/GSA/GGA/GSV must be checked
`transpond_mqtt_list`	array	MQTT forwarding entry list, up to 5 entries, see table below

`transpond_mqtt_list` **entry fields**

Field	Type	Description
`mqtt_name`	string	MQTT connection name, value comes from `mqtt_name_list` of `yruo_service_mqtt`
`topic`	string	Publish topic, up to 511 characters, must not contain special characters
`flag`	integer	Retain flag, `0`=no, `1`=yes
`qos`	integer	QoS level, `0` / `1` / `2`

> The `mqtt_name` + `topic` combination must be unique within the same list.

4.9.4.3 Save GPS MQTT Forwarding Configuration

Request

```
{
  "id": 1,
  "function": "set",
  "core": "yruo_industrial_gps_mqtt",
```

```
  "values": [
    {
      "base": "yruo_industrial_gps_mqtt",
      "type": "yruo_industrial_gps_mqtt",
      "index": 1,
      "value": {
        "enable": 1,
        "trap_interval": 30,
        "RMC": 1,
        "GSA": 1,
        "GGA": 1,
        "GSV": 0,
        "transpond_mqtt_list": [
          { "mqtt_name": "mqtt-conn-1", "topic": "gps/data", "flag": 0, "qos": 0
}
        ]
      }
    }
  ]
}
```

`value` only needs to contain the changed fields. `gps_enable` is a read-only status field and cannot be submitted.

Response

Success: `status` = `0`, `result[0].get` is an empty array.

Failure: `result[0]` contains an error structure:

Field	Type	Description
`templet_code`	integer	Error template code
`params_count`	integer	Number of error parameters
`templet_params`	array	Error parameter list

5. Maintenance

5.1 Tools

5.1.1 Ping Probe

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The result polling interface is a direct `GET` request, with the path returned by the `/cgi` interface.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.1.1.1 Query Ping Status

Called on page load to check whether a Ping task is currently running.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "is_finish",
  "values": [{ "tool": "ping" }]
}
```

Response

When no task is running:

```
{
  "id": 8,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "path": "log/ping_consistent.txt"
        }
      ]
    }
  ]
}
```

When a task is running (`finish` is `0`):

```
{
  "result": [
    {
      "get": [
        {
          "finish": 0,
          "path": "log/ping_consistent.txt",
          "host": "8.8.8.8"
        }
      ]
    }
  ]
}
```

When the task is finished (`finish` is `1`):

```
{
  "result": [
    {
      "get": [
        {
          "finish": 1,
          "path": "log/ping_consistent.txt"
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0]`:

Field	Type	Description
`finish`	number	Ping task status: `0` running, `1` finished; when the field is absent, it is treated as no task running
`path`	string	Relative path of the Ping result log file, used for polling reads
`host`	string	Target host, only present when `finish == 0` (task running)

5.1.1.2 Start Ping

Triggered when the user enters a target host in the input box and clicks the Ping button.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "ping",
  "values": [{ "host": "8.8.8.8" }]
}
```

Parameter	Type	Required	Description
`host`	string	Yes	Target host, IP address or domain name, up to 127 characters

Response

```
{
  "id": 8,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "path": "log/ping_consistent.txt"
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0]`:

Field	Type	Description
`path`	string	Relative path of the Ping result log file, used for polling reads after a successful start

`result[0]` top-level fields:

Field	Type	Description
`error`	any	When present, indicates the start failed

5.1.1.3 Stop Ping

Triggered when the user clicks the stop button; no response callback.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "stop",
  "values": [{ "tool": "ping" }]
}
```

Response

No callback handling; the response content is not parsed.

5.1.1.4 Read Ping Result

The Ping result is stored on the device as a plain-text log file. The frontend polls it once per second via a timer, polling 5 times total, then calls `is_finish` to determine whether it has finished.

Request

```
GET https://<device-ip>/<path>?t=<timestamp>
```

Parameter	Description
`<path>`	Provided by the `path` field in the response of "Start Ping" (5.1.1.2) or "Query Ping Status" (5.1.1.1), e.g. `log/ping_consistent.txt`
`t`	Current millisecond timestamp, used to bypass browser caching

Response

The response is plain text, with the content being the Ping command output and `\n` as the line separator. The frontend replaces the line separators with `<br>` and then displays it in the result area.

5.1.2 Traceroute

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The probe result polling is `GET https://<device-ip>/<path>`

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

Execution Flow

```
Start probe (5.1.2.1)
    └─→ obtain path
           └─→ GET path every second to poll text (5.1.2.2), up to 10 times
                  └─→ after the 10th time, query the completion status (5.1.2.3)
                         ├─→ finish=1: end
                         └─→ finish=0: continue polling
Stop probe (5.1.2.4): can be called at any time to abort the in-progress probe
```

5.1.2.1 Start Traceroute

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "traceroute",
  "values": [{ "host": "8.8.8.8" }]
}
```

Parameter	Type	Description
`host`	string	Target IP or domain name, up to 127 characters

Response

Success:

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
```

```
  "status": 0,
  "result": [
    {
      "get": [
        {
          "path": "log/traceroute_result.txt"
        }
      ]
    }
  ]
}
```

Failure (when `result[0].error` is present):

```
{
  "result": [
    {
      "error": 1,
      "get": []
    }
  ]
}
```

Field Description

Field	Location	Type	Description
`error`	`result[0]`	number	When present and non-zero, indicates the start failed; the frontend shows a tool error prompt
`path`	`result[0].get[0]`	string	Relative path of the probe result file, used for subsequent polling

> The fields in `get[0]` of this interface are directly flattened (no `type` / `index` / `value` wrapping), which differs from the usual `/cgi` convention.

5.1.2.2 Poll Probe Result

After a successful start, the frontend issues a GET request to `path` every 1 second to obtain the probe output text, polling up to 10 times.

Request

```
GET https://<device-ip>/<path>?t=<timestamp>
```

Parameter	Description
`path`	From `result[0].get[0].path` in the 5.1.2.1 response
`t`	Millisecond timestamp, used to prevent browser caching

> This request does not carry the `x-session-id` request header.

Response

The response body is plain text in the traceroute output format; the frontend replaces the line separator `\n` with `<br>` and then renders it on the page.

5.1.2.3 Query Whether Probe Is Finished

After polling 10 times, this interface is called to determine whether the probe has finished; if not finished, polling continues.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "is_finish",
  "values": [{ "tool": "traceroute" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "finish": 1
        }
      ]
    }
  ]
}
```

Field Description

Field	Location	Type	Description
`finish`	`result[0].get[0]`	number	`1` probe finished; `0` still in progress, the frontend continues polling

---

5.1.2.4 Stop Traceroute

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "stop",
  "values": [{ "tool": "traceroute" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [{}]
}
```

> After stopping, the frontend clears the polling timer and resets the button state.

5.1.3 Packet Analyzer

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, with the format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.1.3.1 Get Capture Configuration and Status

Called on page load; returns the current capture configuration, the list of available network interfaces, and the running status.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "get",
  "values": [{ "base": "tcpdump" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "tcpdump",
          "index": 1,
          "value": {
            "interface": ["eth0", "wwan0", "usb0"],
            "selected_interface": "eth0",
            "ip": "",
            "port": "",
            "advanced": "",
            "running": 0,
            "download": 0
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0].value`:

Field	Type	Description
`interface`	array<string>	List of all available network interface names on the device

Field	Type	Description
`selected_interface`	string	The last selected capture interface; when the field is absent, `interface[0]` is used as the default
`ip`	string	IP filter address; an empty string means no filtering
`port`	string | number	Port filter; an empty string means no filtering, and when a value is present it is a number (1–65535)
`advanced`	string	Custom tcpdump parameters for advanced mode; an empty string means advanced mode is not enabled
`running`	number	Capture running status: `0` not running, `1` running
`download`	number	Capture file downloadable status: `0` no file available, non-`0` a file is available for download

5.1.3.2 Start Capture

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "tcpdump_start",
  "values": [
    {
      "interface": "eth0",
      "ip": "192.168.1.100",
      "port": 80,
      "advanced": ""
    }
  ]
}
```

`values[0]` field description:

Field	Type	Required	Description
`interface`	string	Yes	Capture network interface name, taken from the `interface` list in the "Get Configuration" response
`ip`	string	Yes	IP filter address (IPv4); an empty string means no filtering

Field	Type	Required	Description
`port`	string | number	Yes	Port filter (1–65535); pass an empty string `""` when there is no filtering
`advanced`	string	Yes	Custom tcpdump parameters for advanced mode; pass an empty string `""` when not in advanced mode

> Advanced mode (`advanced` non-empty) is mutually exclusive with `ip` / `port` filtering: when advanced mode is enabled, the `ip` / `port` fields are set to read-only by the frontend.

Response

On success there is no structured response; the frontend directly re-calls the page-load interface to refresh the status.

On failure, `result[0]` contains error information:

```
{
  "result": [
    {
      "templet_code": 1234,
      "params_count": 1,
      "templet_params": ["eth0"]
    }
  ]
}
```

`result[0]` top-level error fields:

Field	Type	Description
`templet_code`	number	Error template number
`params_count`	number	Number of error template parameters
`templet_params`	array	Error template parameter list, filled in order into the `{}` placeholders in the template

---

5.1.3.3 Stop Capture

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "tcpdump_stop",
  "values": []
}
```

Response

No structured response; the frontend directly re-calls the page-load interface to refresh the status.

5.1.3.4 Download Capture File

After the capture is complete (`download` non-`0`), the pcap-format file can be downloaded.

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=tcpdump&file=webcapture.pcap
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`type`	Fixed value `tcpdump`
`file`	Fixed value `webcapture.pcap`

Response

The response is a binary stream of the pcap-format file. Example response headers:

```
Content-Disposition: attachment; filename="webcapture.pcap"
Content-Type: application/octet-stream
```

The filename is taken from `Content-Disposition: attachment; filename="..."`, falling back to `webcapture.pcap`.

5.1.4 Qxdmlog

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The log download interface path is `POST https://<device-ip>/cgi-bin/qxdm-export`, with the format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model

Field	Type	Description
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.1.4.1 Get Qxdmlog Status

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "qxdm_get",
  "values": [{ "base": "qxdm" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "qxdm",
          "index": 0,
          "value": {
            "running": 0,
            "download": 0
          }
        }
      ]
    }
  ]
}
```

Field Description

`get[0]` value object:

Field	Type	Description
`running`	number	Collection status: `1` collecting, `0` not collecting
`download`	number	Log file availability: `0` no file available for download, `1` a file is available for download (the frontend also displays a "Please stop collection first" prompt)

5.1.4.2 Start Qxdmlog Collection

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "qxdm_start",
  "values": []
}
```

Response

Success: `result[0]` does not contain `templet_code`; the frontend automatically refreshes the status.

Failure:

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "templet_code": 10001001,
      "params_count": 0,
      "templet_params": []
    }
  ]
}
```

Field Description

On failure, the error fields are located at the `result[0]` top level (not inside `get[]`):

Field	Type	Description
`templet_code`	number	Error code; the frontend looks up the corresponding prompt text based on this
`params_count`	number	Number of `{}` placeholders in the text template

Field	Type	Description
`templet_params`	array<string>	Parameter values that replace the placeholders in order

5.1.4.3 Stop Qxdmlog Collection

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_tools",
  "function": "qxdm_stop",
  "values": []
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [{}]
}
```

> After a successful stop, the frontend automatically refreshes the status page.

5.1.4.4 Download Qxdmlog File

> Note: This interface path is `/cgi-bin/qxdm-export`, not the common `/cgi-bin/file-export`.

Request

```
POST https://<device-ip>/cgi-bin/qxdm-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=qxdm-export&file=webcapture.pcap
```

Parameter	Type	Description
`sessionid`	string	Session credential, the same as the `x-session-id` request header value

Parameter	Type	Description
`type`	string	Fixed value `qxdm-export`
`file`	string	Fixed value `webcapture.pcap`

Response

The response body is the binary content of the `.pcap` file, with the filename fixed to `webcapture.pcap`.

> The download button is only available when `running=0` and `download=1`; while collection is in progress, you must stop it before downloading.

5.2 Debug

5.2.1 Cellular AT Debug

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, with the format `application/x-www-form-urlencoded`.
> The log clear interface path is `POST https://<device-ip>/cgi-bin/file-delete`, with the format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.2.1.1 Send AT Command / Get AT Log

Sending an AT command and refreshing the log share the same interface. When `command` is empty, only the current log is refreshed; when non-empty, the command is executed first and then the latest log is returned.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_debug_at",
  "function": "get",
  "values": [
    {
      "base": "at_log",
      "command": "AT+CGREG?",
      "line": 20
    }
  ]
}
```

`values[0]` field description:

Field	Type	Required	Description
`base`	string	Yes	Fixed value `at_log`
`command`	string	Yes	AT command string, up to 255 characters; an empty string means only refreshing the log without sending a command
`line`	number	Yes	Number of log lines to return; selectable values: `20` / `50` / `100` / `1000`

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "value": {
            "log": "AT+CGREG?\r\n+CGREG: 0,1\r\nOK\r\n"
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0].value`:

Field	Type	Description
`log`	string	AT log content, including command echo and device response; an empty string means no log yet

---

5.2.1.2 Download AT Log

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=log_at&file=log_at
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`type`	Fixed value `log_at`
`file`	Fixed value `log_at`

Response

The response is a file binary stream; the filename is taken from the `Content-Disposition` response header, falling back to `log_at`.

5.2.1.3 Clear AT Log

Request

```
POST https://<device-ip>/cgi-bin/file-delete
Content-Type: application/x-www-form-urlencoded
```

Request body:

```
sessionid=<td>&type=log_at&file=log_at
```

Field	Description
`sessionid`	Session credential
`type`	Fixed value `log_at`

Field	Description
`file`	Fixed value `log_at`

Response

No structured response; after a successful operation, the frontend clears the log display area.

5.2.2 Firewall

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, with the format `application/x-www-form-urlencoded`.
> The log clear interface path is `POST https://<device-ip>/cgi-bin/file-delete`, with the format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.2.2.1 Execute Firewall Command / Get Firewall Log

Executing an iptables command and refreshing on page load share the same interface. The `command` field is only passed when executing a command, and is not passed when refreshing. `line` is fixed to `1000`.

Request

When executing a command:

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_debug_firewall",
  "function": "get",
  "values": [
    {
      "base": "firewall_log",
      "command": "-t nat -nvL INPUT",
      "line": 1000
    }
  ]
}
```

When refreshing / page load:

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_debug_firewall",
  "function": "get",
  "values": [
    {
      "base": "firewall_log",
      "line": 1000
    }
  ]
}
```

`values[0]` field description:

Field	Type	Required	Description
`base`	string	Yes	Fixed value `firewall_log`
`command`	string	No	iptables command parameters, up to 255 characters; only passed when executing a command, omitted when refreshing
`line`	number	Yes	Fixed value `1000`

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
```

```
          "value": {
            "log": "Chain INPUT (policy ACCEPT)\ntarget  prot opt source
destination\n"
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0].value`:

Field	Type	Description
`log`	string	Firewall log content, including iptables command output; an empty string means no log yet

---

5.2.2.2 Download Firewall Log

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=log_firewall&file=log_firewall
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`type`	Fixed value `log_firewall`
`file`	Fixed value `log_firewall`

Response

The response is a file binary stream. Example response headers:

```
Content-Disposition: attachment; filename="log_firewall"
Content-Type: application/octet-stream
```

The filename is taken from `Content-Disposition: attachment; filename="..."`, falling back to `log_firewall`.

5.2.2.3 Clear Firewall Log

Request

```
POST https://<device-ip>/cgi-bin/file-delete
Content-Type: application/x-www-form-urlencoded
```

Request body:

```
sessionid=<td>&type=log_firewall&file=log_firewall
```

Field	Description
`sessionid`	Session credential
`type`	Fixed value `log_firewall`
`file`	Fixed value `log_firewall`

Response

No structured response; after a successful operation, the frontend clears the log display area.

5.3 Log

5.3.1 System Log

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The log clear interface path is `POST https://<device-ip>/cgi-bin/file-delete`, with the format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.3.1.1 Get System Log

Called on page load, when switching the number of displayed lines, or on auto-refresh.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_log",
  "function": "get",
  "values": [{ "base": "system_log", "line": 20 }]
}
```

Parameter	Type	Required	Description
`line`	number	Yes	Number of log lines to return; selectable values: `20` / `50` / `100` / `1000`

Response

```
{
  "id": 30,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "system_log",
          "index": 1,
          "value": {
            "log": "Fri Apr 10 04:48:20 2026 daemon.info dhcpd: Sending on
Socket/fallback/fallback-net\nFri Apr 10 04:48:20 2026 daemon.info dhcpd: Server
starting service.\n..."
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0].value`:

Field	Type	Description
`log`	string	System log content, multiple lines separated by `\n`; an empty string or field absence when there is no log

The format of each log line is:

```
<weekday> <month> <day> <time> <year> <facility>.<level> <process>: <message>
```

Example: `Fri Apr 10 04:48:20 2026 daemon.info dhcpd: Server starting service.`

5.3.1.2 Clear System Log

Request

```
POST https://<device-ip>/cgi-bin/file-delete
Content-Type: application/x-www-form-urlencoded
```

Request body:

```
sessionid=<td>&type=log&file=system.log
```

Field	Description
`sessionid`	Session credential
`type`	Fixed value `log`
`file`	Fixed value `system.log`

Response

```
{ "delete": 0 }
```

Field	Type	Description
`delete`	number	`0` clear succeeded, non-`0` clear failed

5.3.2 System Log Download

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, with the format `application/x-www-form-urlencoded`.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)

Field	Type	Description
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.3.2.1 Get Log File List

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_log",
  "function": "get",
  "values": [{ "base": "log_list" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "log_list",
          "index": 1,
          "value": {
            "list": [
              { "name": "syslog.1", "size": "1.2 MB", "time": "2026-04-10
04:48:20" },
              { "name": "syslog.2", "size": "0.8 MB", "time": "2026-04-09
04:48:20" }
            ]
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0].value`:

Field	Type	Description
`list`	array	Log file list

`list[]` elements:

Field	Type	Description
`name`	string	Log file name, used as the `file` parameter when downloading
`size`	string	File size, including the unit
`time`	string	File creation time

5.3.2.2 Download a Single Log File

Triggered by the download button in the action column of each table row; `file` takes the `name` field value of the corresponding row.

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=log&file=<filename>
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`type`	Fixed value `log`
`file`	Log file name, taken from the `name` field of the corresponding row in the log list

Response

The response is a file binary stream. Example response headers:

```
Content-Disposition: attachment; filename="syslog.1"
Content-Type: application/octet-stream
```

The filename is taken from `Content-Disposition: attachment; filename="..."`, falling back to the `file` field value in the request.

5.3.2.3 Download All Logs

Triggered by the "Download All" button at the top of the page; downloads all logs as a package.

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=log&file=all
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`type`	Fixed value `log`
`file`	Fixed value `all`, indicating downloading all logs

Response

The response is a binary stream of an archive-format file. Example response headers:

```
Content-Disposition: attachment; filename="all"
Content-Type: application/octet-stream
```

The filename is taken from `Content-Disposition: attachment; filename="..."`, falling back to `all`.

5.3.3 System Log Settings

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.3.3.1 Get Log Settings

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_log",
  "function": "get",
  "values": [{ "base": "log_settings" }]
}
```

Response

```
{
  "id": 86,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "available_storage": ["local"],
      "get": [
        {
          "type": "log_settings",
          "index": 1,
          "value": {
            "remote_enable": 0,
            "console_enable": 0,
            "storage": "local",
            "size": 2048,
            "loglevel": 7
          }
        }
      ]
    }
  ]
}
```

When remote logging is enabled (`remote_enable: 1`), `value` additionally contains:

```
{
  "remote_enable": 1,
  "remote_host": "192.168.1.100",
  "remote_port": 514,
  ...
}
```

Field Description

`result[0]` top-level fields:

Field	Type	Description
`available_storage`	array<string>	List of available storage media on the device, used as the valid value range for the `storage` field; known values: `local` (local) / `Micro SD` / `SSD`

`result[0].get[0].value`:

Field	Type	Description
`remote_enable`	number	Whether remote logging is enabled: `0` off, `1` on
`remote_host`	string	Remote syslog server address (IP or domain name), only present when `remote_enable: 1`
`remote_port`	number	Remote syslog port (0–65535), only present when `remote_enable: 1`, default `514`
`console_enable`	number	Whether console logging is enabled: `0` off, `1` on (the frontend currently does not display this field)
`storage`	string	Log storage medium, the value comes from `available_storage`; known values: `local` / `Micro SD` / `SSD`
`size`	number	Log file size limit (KB); maximum `5120` for `local` storage, maximum `512000` for `Micro SD` / `SSD`
`loglevel`	number	Log level, `1`–`8` (larger values record more detail); known enum values, possibly incomplete

---

5.3.3.2 Save Log Settings

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_log",
  "function": "set",
  "values": [
    {
      "base": "log_settings",
      "type": "log_settings",
      "index": 1,
      "value": {
        "storage": "local",
        "size": 4096,
        "loglevel": 5
```

```
      }
    }
  ]
}
```

`values[0].value` only contains the changed fields.

Response

On success, the top-level `status: 0`; `result[0].get` is an empty array or symmetric with the GET structure.

On failure, `result[0]` contains error information:

```
{
  "result": [
    {
      "templet_code": 1234,
      "params_count": 0,
      "templet_params": []
    }
  ]
}
```

Field	Type	Description
`templet_code`	number	Error template number
`params_count`	number	Number of error template parameters
`templet_params`	array	Error template parameter list, filled in order into the `{}` placeholders in the template

---

5.4 Upgrade

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with the format `multipart/form-data`.
> The force upgrade interface path is `POST https://<device-ip>/cgi-bin/force_upgrade` or `POST https://<device-ip>/cgi-bin/force_upgrade_reset`, with no request body.
> The cancel upgrade interface path is `POST https://<device-ip>/cgi-bin/cancel_upgrade`.

Top-level Response Fields (common to all /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model

Field	Type	Description
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.4.1 Get Upgrade Information

Called on page load and when a UPS file is selected; returns the current firmware version and UPS connection status.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_upgrade",
  "function": "get",
  "values": [{ "base": "upgrade" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "upgrade",
          "index": 1,
          "value": {
            "cur_firmware": "35.3.0.12-a1",
            "withreset": 0,
            "ups_power_status": "2",
            "ups_firm_ver": "1.0.0"
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0].value`:

Field	Type	Description
`cur_firmware`	string	Current device firmware version
`withreset`	number	Whether "Restore Factory Settings" is checked by default: `0` no, `1` yes
`ups_power_status`	string	UPS power status, only returned when the UPS hardware is present; `"1"` UPS connected (on battery), `"2"` UPS connected (external power)
`ups_firm_ver`	string	Current UPS firmware version, only returned when `ups_power_status` is present

---

5.4.2 Upload Firmware

Upload a `.bin` format firmware file to trigger the upgrade process. The `withreset` checkbox determines the upload type:

- "Restore Factory Settings" not checked: `type=upgrade`
- "Restore Factory Settings" checked: `type=upgrade_with_reset`

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
x-session-id: <td>
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`filename`	When not checked: `type=upgrade&file=firmware.bin`; when "Restore Factory Settings" is checked: `type=upgrade_with_reset&file=firmware.bin`
`size`	File size (bytes)
`file`	Firmware file binary content (`.bin` format)

Response

Success:

```
{ "size": 12582912, "checksum": "a1b2c3d4", "filename": "firmware.bin" }
```

Failure (containing `templet_code`):

```
{ "templet_code": 50201001, "params_count": 0, "templet_params": [] }
```

> `templet_code: 50201001` indicates that the uploaded firmware is the same as the current version; the frontend will pop up a confirmation box, allowing the user to choose to force the upgrade (see 5.4.3) or cancel (see 5.4.4).

5.4.3 Force Upgrade

Triggered when the user clicks confirm in the confirmation box, after the uploaded firmware is the same as the current version (`templet_code: 50201001`).

Request

Without restoring factory settings:

```
POST https://<device-ip>/cgi-bin/force_upgrade
```

With restoring factory settings:

```
POST https://<device-ip>/cgi-bin/force_upgrade_reset
```

No request body.

Response

The structure is the same as the file upload response:

Success: `{ "size": <number>, "checksum": "<string>", "filename": "<string>" }`

Failure: `{ "templet_code": <number>, "params_count": <number>, "templet_params": [...] }`

5.4.4 Cancel Upgrade

Triggered when the user clicks cancel in the same-version confirmation box.

Request

```
POST https://<device-ip>/cgi-bin/cancel_upgrade
```

Request body (JSON):

```
{ "core": "", "function": "", "values": [] }
```

Response

No response handling.

5.4.5 Upload UPS Firmware

Only displayed when `ups_power_status` is `"1"` or `"2"` (UPS connected). Before uploading, `ups_power_status` is queried again; if `ups_power_status != "2"` (no external power connected), the upload is rejected.

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
x-session-id: <td>
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`filename`	Fixed value `type=ups_upgrade&file=ups_firmware.bin`
`size`	File size (bytes)
`file`	UPS firmware file binary content (`.bin` format)

Response

Success: `{ "size": <number>, "checksum": "<string>", "filename": "<string>" }`

Failure `message` field value meanings (known enum values, possibly incomplete):

`message`	Description
`1`	Error type 1
`2`	Error type 2
`3`	Error type 3 (firmware verification failed)

5.4.6 Trigger UPS Upgrade

Called immediately after the file is uploaded successfully, starting the UPS firmware upgrade process.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_upgrade",
  "function": "set",
  "values": [{ "base": "ups_upgrade" }]
}
```

Response

No structured response; the frontend then calls 5.4.7 to poll the upgrade status.

5.4.7 Query UPS Upgrade Status

Queried once immediately after the file is uploaded successfully; if an upgrade is in progress, it is polled every 5 seconds until the upgrade completes or fails.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_upgrade",
  "function": "get",
  "values": [{ "base": "ups_upgrade" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "value": {
            "ups_upgrade_result": "1"
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get[0].value`:

Field	Type	Description
`ups_upgrade_result`	string	UPS upgrade result: `"0"` success, `"1"` upgrading, `"2"` failure, `"3"` firmware verification failed

---

5.5 Backup & Restore

> The `/cgi` interface uniformly uses `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with the format `multipart/form-data`.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, with the format `application/x-www-form-urlencoded`.

> **Note**: The response examples in this document are all inferred from the code and have not been verified against actual interface returns.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.5.1 Restore Configuration

Upload a `.dat` format configuration backup file. After a successful upload, the device will automatically apply the configuration and trigger logout after about 5 seconds.

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
x-session-id: <td>
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`filename`	Fixed value `type=backup&file=cfgbackup`
`size`	File size (bytes)
`file`	Configuration backup file binary content (`.dat` format)

Response

Success:

```
{ "size": 4096, "checksum": "a1b2c3d4", "filename": "cfgbackup" }
```

Failure:

```
{ "message": 1 }
```

`message`	Description
`1`	File verification failed (firmware/configuration mismatch)

5.5.2 Backup Configuration

Download the complete configuration file of the current device.

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=backup&file=cfgbackup
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`type`	Fixed value `backup`
`file`	Fixed value `cfgbackup`

Response

The response is a file binary stream. Example response headers:

```
Content-Disposition: attachment; filename="cfgbackup.dat"
Content-Type: application/octet-stream
```

The filename is taken from `Content-Disposition: attachment; filename="..."`, falling back to `cfgbackup.dat`.

5.5.3 Batch Backup

Download a configuration backup file suitable for batch deployment (not containing device-specific information).

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=batch_backup&file=batch_cfgbackup
```

Field	Description
`sessionid`	Session credential, the same as the `x-session-id` request header value
`type`	Fixed value `batch_backup`
`file`	Fixed value `batch_cfgbackup`

Response

The response is a file binary stream. Example response headers:

```
Content-Disposition: attachment; filename="batch_cfgbackup.dat"
Content-Type: application/octet-stream
```

The filename is taken from `Content-Disposition: attachment; filename="..."`, falling back to `batch_cfgbackup.dat`.

5.5.4 Restore Factory Settings

Triggered when the user clicks confirm in the confirmation box. The device will clear all configurations and reboot; the frontend waits about 20 seconds and then redirects to the login page.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_upgrade",
  "function": "reset",
  "values": [{ "base": "default_cfg" }]
}
```

Response

No callback handling; the response content is not parsed.

5.6 Reboot

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to all interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version
`status`	number	Request execution status: `0` success

5.6.1 Get Scheduled Reboot Configuration

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_schedule",
  "function": "get",
  "values": [{ "base": "schedule" }]
}
```

Response

```
{
  "id": 12,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "servs": ["reboot"],
      "get": [
        {
```

```
          "type": "schedule",
          "index": 1,
          "value": {
            "enable": 0,
            "type": 0,
            "day": 7,
            "hour": 0,
            "minute": 0
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0]` top-level fields:

Field	Type	Description
`servs`	array<string>	List of services controlled by this configuration, currently fixed to `["reboot"]`

`result[0].get[0].value`:

Field	Type	Description
`enable`	number	Whether scheduled reboot is enabled: `0` off, `1` on
`type`	number	Reboot cycle type: `0` daily, `1` weekly, `2` monthly
`day`	number	Execution date: when `type=1`, the day of the week (`1`–`7`, `1`=Monday, `7`=Sunday); when `type=2`, the day of the month (`1`–`31`); when `type=0`, the field is present but has no actual meaning
`hour`	number	Execution hour (`0`–`23`)
`minute`	number	Execution minute (`0`–`59`)

5.6.2 Save Scheduled Reboot Configuration

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_schedule",
  "function": "set",
  "values": [
    {
      "base": "schedule",
      "type": "schedule",
```

```
      "index": 1,
      "value": {
        "enable": 1,
        "type": 1,
        "day": 1,
        "hour": 3,
        "minute": 0
      }
    }
  ]
}
```

`values[0].value` only contains the changed fields; the field meanings are the same as in the GET response.

Response

On success, the top-level `status: 0`.

On failure, `result[0]` contains error information:

```
{
  "result": [
    {
      "templet_code": 1234,
      "params_count": 0,
      "templet_params": []
    }
  ]
}
```

5.6.3 Reboot Now

Triggered when the user clicks confirm in the confirmation box. The device will reboot immediately; the frontend waits about 35 seconds and then redirects to the login page.

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_upgrade",
  "function": "reboot",
  "values": [{}]
}
```

Response

No callback handling; the response content is not parsed.

6. APP

6.1 Python

6.1.1 Python

> All `/cgi` interfaces use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with `multipart/form-data` format; see each section for details.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version number
`status`	number	Request execution status: `0` means success

6.1.1.1 Get Python SDK Status

Request

```
{
  "id": 5,
  "execute": 1,
  "core": "yruo_python",
  "function": "get",
  "values": [{ "base": "status" }]
}
```

Response

```
{
  "id": 5,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "available_storage": [],
      "match": 1,
```

```
      "get": [
        {
          "type": "status",
          "index": 0,
          "value": {
            "sdk_version": "",
            "appmanager_state": "Uninstalled",
            "sdk_path": ""
          }
        }
      ]
    }
  ]
}
```

Field Description

In addition to the standard `get` array, `result[0]` contains the following top-level fields:

Field	Type	Description
`available_storage`	array<string>	List of storage paths available for installing the SDK; empty array when no storage is available
`match`	number	Version compatibility: `1` means version matches, `0` means version mismatch (the front end will display an incompatibility warning)

`get[0]` (type = `"status"`) value object:

Field	Type	Description
`sdk_version`	string	Installed Python SDK version number; empty string when not installed
`appmanager_state`	string	AppManager running state; enum values are listed in the table below
`sdk_path`	string	SDK installation path; empty string when not installed

`appmanager_state` enum values:

Value	Description
`"Uninstalled"`	SDK not installed
`"Running"`	SDK installed and AppManager is running (front end shows a clickable link, port `9001`)

6.1.1.2 Install Python SDK

Upload the SDK package to a specified storage path on the device and install it. Before uploading, first obtain the `available_storage` list via `6.1.1.1` and select a target path.

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
x-session-id: <td>
```

Form fields:

Field	Type	Description
`sessionid`	string	Session credential, same as the `x-session-id` request header value
`filename`	string	Fixed format `type=pythonsdk&file=<storage_path>`, where `storage_path` is a path selected from `available_storage[]`
`size`	string	File size (bytes)
`file`	binary	SDK package file binary content

Response

Success:

```
{
  "size": 1234567,
  "checksum": "a1b2c3d4e5f6...",
  "filename": "pythonsdk.tar.gz"
}
```

Failure:

```
{
  "templet_code": 10001001,
  "params_count": 0,
  "templet_params": []
}
```

Field Description

Success response:

Field	Type	Description
`size`	number	Uploaded file size (bytes)
`checksum`	string	File checksum value
`filename`	string	File name saved on the server

Failure response (treated as failure when both `size` and `checksum` are missing):

Field	Type	Description
`templet_code`	number	Error code; the front end looks up the corresponding prompt text based on this
`params_count`	number	Number of `{}` placeholders in the text template
`templet_params`	array<string>	Parameter values that replace the placeholders in order

> After a successful upload, the front end automatically refreshes the status page; while uploading, if `available_storage` is empty or the SDK is already installed, the install button is unavailable.

6.1.1.3 Uninstall Python SDK

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_python",
  "function": "uninstall",
  "values": [{ "base": "python_sdk" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [{}]
}
```

> After a successful uninstall, `status` is `0` and the `result` content is an empty object. The front end automatically refreshes the status page after the response is returned.

6.1.2 AppManager Config

> All requests use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version number
`status`	number	Request execution status: `0` means success

6.1.2.1 Get AppManager Config

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_python",
  "function": "get",
  "values": [{ "base": "appmanager" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "get": [
        {
          "type": "appmanager",
          "index": 0,
          "value": {
            "enable": 1,
            "python_installed": 1
          }
        },
        {
          "type": "appconfig",
          "index": "0",
          "value": {
            "id": "app1",
            "command": "python3 main.py",
```

```
            "log_size": 10
          }
        },
        {
          "type": "appconfig",
          "index": "1",
          "value": {
            "id": "app2",
            "command": "python3 run.py",
            "log_size": 5
          }
        },
        // ... more appconfig entries
        {
          "type": "appinfo",
          "index": "0",
          "value": {
            "name": "app1",
            "app_version": "1.0.0",
            "sdk_version": "3.8.0"
          }
        },
        {
          "type": "appinfo",
          "index": "1",
          "value": {
            "name": "app2",
            "app_version": "2.1.3",
            "sdk_version": "3.8.0"
          }
        },
        // ... more appinfo entries
        {
          "type": "account_info",
          "index": 0,
          "value": {
            "username": "admin"
          }
        }
      ]
    }
  ]
}
```

Field Description

`result[0].get` contains four types of data, distinguished by `type`:

- `appmanager`: AppManager global switch and installation status; fixed single entry
- `appconfig`: Registered Python APP config list, one entry per APP
- `appinfo`: Installed Python APP info list, one entry per APP
- `account_info`: AppManager Web console account info; fixed single entry

`appmanager` value object

Field	Type	Description
`enable`	number	Whether AppManager is enabled: `1` enabled, `0` disabled
`python_installed`	number	Whether the Python SDK is installed: `1` installed, `0` not installed; when not installed, all operation buttons on the front end are set to read-only

`appconfig` value object (APP config)

Field	Type	Description
`id`	string	APP identifier name
`command`	string	APP startup command
`log_size`	number	Log file size limit (MB), range `1`–`50`

`appinfo` value object (APP info)

Field	Type	Description
`name`	string	APP name
`app_version`	string	APP version number
`sdk_version`	string	Python SDK version the APP depends on

`account_info` value object (console account)

Field	Type	Description
`username`	string	AppManager Web console login username (read-only)

> Password fields (`old_passwd` / `new_passwd` / `confirm_passwd`) are only uploaded on save; they are not returned in the GET response.

6.1.2.2 Save AppManager Config

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_python",
  "function": "set",
  "values": [
    {
      "base": "appmanager",
      "type": "appmanager",
      "index": 0,
```

```
      "value": { "enable": 1 }
    },
    {
      "base": "appmanager",
      "type": "appconfig",
      "index": "0",
      "value": { "log_size": 20 }
    },
    {
      "base": "appmanager",
      "type": "account_info",
      "index": 0,
      "value": {
        "username": "admin",
        "old_passwd": "oldPassword1",
        "new_passwd": "newPassword1",
        "confirm_passwd": "newPassword1"
      }
    }
  ]
}
```

> `values` only contains entries that have changed; when modifying the password, `old_passwd`, `new_passwd`, and `confirm_passwd` must be sent together.

Response

On save success: `result[0]` is an empty object `{}`.

On save failure, `result[0]` contains error info:

```
{
  "result": [
    {
      "templet_code": 60102001,
      "params_count": 1,
      "templet_params": ["admin"]
    }
  ]
}
```

Field	Type	Description
`templet_code`	number	Error code; the front end looks up the corresponding prompt text based on this
`params_count`	number	Number of `{}` placeholders in the text template
`templet_params`	array<string>	Parameter values that replace the placeholders in the text in order

6.1.2.3 Uninstall a Specified Python APP

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_python",
  "function": "app_uninstall",
  "values": [
    {
      "base": "app_uninstall",
      "index": "0",
      "value": { "name": "app1" }
    }
  ]
}
```

Parameter	Type	Description
`index`	string	Corresponds to the `index` of the `appconfig` entry
`value.name`	string	Name of the APP to uninstall (corresponds to `appconfig.value.id`)

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN00000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [{}]
}
```

> After a successful uninstall, the front end automatically refreshes the AppManager config page.

6.1.3 Python APP

> All `/cgi` interfaces use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.
> The file upload interface path is `POST https://<device-ip>/cgi-bin/file-import`, with `multipart/form-data` format.
> The file export interface path is `POST https://<device-ip>/cgi-bin/file-export`, with `application/x-www-form-urlencoded` format.

Top-level Response Fields (common to /cgi interfaces)

Field	Type	Description
`id`	number	Request ID, corresponding to the `id` in the request
`model`	string	Device model
`pn`	string	Product number (PN)
`oem`	string	OEM identifier
`rtver`	string	Firmware version number
`status`	number	Request execution status: `0` means success

6.1.3.1 Get Python APP List

Request

```
{
  "id": 1,
  "execute": 1,
  "core": "yruo_python",
  "function": "get",
  "values": [{ "base": "python_apps" }]
}
```

Response

```
{
  "id": 1,
  "model": "UR35",
  "pn": "L04EU2211110CN0000000000",
  "oem": "0000",
  "rtver": "35.3.0.12-a1",
  "status": 0,
  "result": [
    {
      "installed_apps": ["app1", "app2"],
      "python_installed": 1,
      "script_files": ["debug.py", "test.py"],
      "get": []
    }
  ]
}
```

Field Description

In addition to the standard `get` array, `result[0]` contains the following top-level fields:

Field	Type	Description
`installed_apps`	array<string>	List of installed Python APP names; empty array when no APP is installed
`python_installed`	number	Whether the Python SDK is installed: `1` installed, `0` not installed; when not installed, all operations on the front end are read-only
`script_files`	array<string>	List of available debug script file names on the device; empty array when there are no scripts

> The `get` array for this interface is always empty; all data needed by the page comes from the top-level fields of `result[0]`.

6.1.3.2 Install Python APP

Upload a Python APP installation package (`.tar` / `.zip`) to the device.

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
x-session-id: <td>
```

Field	Type	Description
`sessionid`	string	Session credential, same as the `x-session-id` request header value
`filename`	string	Fixed value `type=pythonapp&file=package`
`size`	string	File size (bytes)
`file`	binary	APP package file binary content (`.tar` / `.zip`)

Response

The success and failure response structure is the same as 6.1.1.2 Install Python SDK.

> A confirmation dialog pops up before uploading; after a successful upload, the front end automatically refreshes the APP list.

6.1.3.3 Upload APP Config File

Upload a config file for a specified installed APP.

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
x-session-id: <td>
```

Field	Type	Description
`sessionid`	string	Session credential, same as the `x-session-id` request header value
`filename`	string	Format `type=pythonapp_cfg&file=<app_name>`, where `app_name` is an APP name selected from `installed_apps[]`
`size`	string	File size (bytes)
`file`	binary	Config file binary content

Response

The success and failure response structure is the same as 6.1.1.2 Install Python SDK.

6.1.3.4 Upload Debug Script

Upload a Python script file used for debugging to the device.

Request

```
POST https://<device-ip>/cgi-bin/file-import
Content-Type: multipart/form-data
x-session-id: <td>
```

Field	Type	Description
`sessionid`	string	Session credential, same as the `x-session-id` request header value
`filename`	string	Fixed value `type=pyscript_in&file=inscript`
`size`	string	File size (bytes)
`file`	binary	Python script file binary content

Response

The success and failure response structure is the same as 6.1.1.2 Install Python SDK.

> After a successful upload, the front end automatically refreshes the page; the script file name will appear in the `script_files[]` list.

6.1.3.5 Export Debug Script

Download the specified debug script file from the device.

Request

```
POST https://<device-ip>/cgi-bin/file-export
Content-Type: application/x-www-form-urlencoded
x-session-id: <td>
```

Request body:

```
sessionid=<td>&type=pyscript_out&file=<filename>
```

Parameter	Type	Description
`sessionid`	string	Session credential, same as the `x-session-id` request header value
`type`	string	Fixed value `pyscript_out`
`file`	string	Script file name, taken from `script_files[]` in the `6.1.3.1` response; when `script_files` is empty, the export button is unavailable

Response

The response body is the binary content of the script file; the file name is taken from the `Content-Disposition` response header, falling back to the `file` parameter value from the request.

7. Other

7.1 Login

7.1.1 Login

Request

`POST https://<device-ip>/cgi`, **no** `x-session-id` request header required.

```
{
  "id": "1",
  "execute": 1,
  "core": "user",
  "function": "login",
  "values": [
    {
      "username": "admin",
      "password": "<encrypted>"
    }
  ]
}
```

Password Encryption

The password must be AES-CBC encrypted before being sent:

Parameter	Value
Algorithm	AES-CBC

Parameter	Value
Key	Key
IV	Offset
Padding	PKCS7
Output format	Convert the ciphertext hex string to uppercase, then perform Base64 encoding

Example (CryptoJS):

```
var encrypted = CryptoJS.AES.encrypt(CryptoJS.enc.Utf8.parse(password), key, {
  iv: iv,
  mode: CryptoJS.mode.CBC,
  padding: CryptoJS.pad.Pkcs7,
});
var encryptedPassword = CryptoJS.enc.Base64.stringify(
  CryptoJS.enc.Hex.parse(encrypted.ciphertext.toString().toUpperCase())
);
```

Response

Login Successful (`status=0`)

```
{
  "status": 0,
  "result": [
    {
      "td": "<session-token>",
      "ysrole": 1
    }
  ]
}
```

Field	Type	Description
`td`	string	Session credential; all subsequent requests must carry `x-session-id: <td>` in the request header
`ysrole`	number	User role, `0`=read-only, `1`=read-write

Incorrect Username or Password (`status=-1`)

```
{ "status": -1 }
```

Login Restricted (`status=-2`)

```
{
  "status": -2,
  "result": [
    {
      "locktime": 300,
      "chance": 2,
      "change": 0
    }
  ]
}
```

Field	Type	Description
`locktime`	number	Remaining account lock time (seconds); `0` indicates not locked but the number of remaining attempts is limited
`chance`	number	Remaining number of attempts (valid when `locktime=0`)
`change`	number	Whether the password needs to be changed, `1`=yes, `0`=no

---

7.1.2 Check Login Status

Request

`POST https://<device-ip>/islogin`, no request body, no `x-session-id` request header required.

Response

```
{
  "status": 0,
  "oem": "0000",
  "model": "UR35"
}
```

Field	Type	Description
`status`	number	`0`=logged in (will redirect to the home page), non-`0`=not logged in
`oem`	string	OEM identifier
`model`	string	Device model

7.1.3 Logout

Request

`POST https://<device-ip>/logout`, carries the session credential via the `x-session-id: <td>` request header.

```
{ "core": "", "function": "", "value": "" }
```

Response

There is no fixed response body. After the request succeeds, the server destroys the session. The client should clear the locally stored session credential and redirect to the login page.

7.2 Application Config

> All requests uniformly use `POST https://<device-ip>/cgi`, carrying the session credential via the `x-session-id: <td>` request header.

7.2.1 Global Application

After the user modifies configuration on the page, they need to click the "Apply" button in the top navigation bar to send all pending changes to the device at once.

Request

```
{
  "core": "yruo_apply",
  "function": "apply",
  "values": []
}
```

Response

Apply Successful (`status=0`)

```
{
  "status": 0,
  "result": [
    {
      "reboot": 0,
      "reconnect": 0
    }
  ]
}
```

Field	Type	Description
`reboot`	number	Whether the device needs to be rebooted, `1`=reboot required, `0`=no reboot required

Field	Type	Description
`reconnect`	number	Whether re-login is required, `1`=required, `0`=not required

Apply Failed (`status=-1`)

```
{ "status": -1 }
```

Session Expired (`status=-2`)

```
{ "status": -2 }
```

> When `status=-2` is received, the frontend will automatically redirect to the login page.

7.3 Error Code

In all interface responses, a non-zero `code` field indicates an error. The frontend uses the `code` mapping table in `zh-cn.js` / `en.js` to convert error codes into user-readable messages.

Placeholder notes: `{}` indicates a parameter value dynamically filled by the backend, and `$` indicates the original error message passed through directly.

General Error Codes (1000xxxx)

Error Code	Chinese Description	English Description
10000000	保存成功	Saved successfully
10000001	未知错误: {}	Unknown error: {}
10000002	参数 {} 无效，原因: {}	Invalid parameter {}, reason: {}
10000003	I/O 错误	I/O error
10000004	内存不足	Not enough storage
10000005	IP 地址 {} 冲突	IP Address {} conflict
10000006	接口名称 {} 不可用	Interface name {} is unavailable
10000007	端口 {} 已被占用	Port {} has been occupied
10000008	用户名 {} 已被占用	Username {} has been occupied
10000009	密码错误	Wrong password
10000010	错误代码: $	Error code: $
10000011	错误代码: $	Error code: $
10000012	错误代码: $	Error code: $

Error Code	Chinese Description	English Description
10000013	无效 IP 地址 {}	Invalid IP address {}
10000014	接口 {} 不存在	Interface {} does not exist
10000015	请先配置串口为 {} 模式	Please configure the serial port for the {} mode first
10000016	请先关闭 {} 功能	Please turn off the {} function first
10000017	网络无法连接	Failed to locate the remote host. Please double check router's network status
10000018	非法文件	Illegal file
10000019	空间不足，请清除数据	Insufficient space, please clear the data
10000020	操作超时	Operation timeout
10000021	IP 地址 {} 与网关 {} 必须同网段	IP address {} and gateway {} must be on the same network segment
10000022	{} 正在引用此号码组，请先取消引用方可删除	{} is referring to this number group, please dereference first to delete
10000023	{} 正在引用此邮箱群组，请先取消引用方可删除	{} is referring to this email group, please dereference first to delete
10000024	{} 正在引用此 SLA，请先取消引用方可删除	{} is referring to this SLA, please dereference first to delete
10000025	{} 正在引用此 TRACK，请先取消引用方可删除	{} is referring to this TRACK, please dereference first to delete
10000026	文件格式不正确	Invalid file format
10000027	请先开启 {} 功能	Please turn on the {} function first
10000028	启用待机模式，点击【应用】10min 后路由器将进入待机模式	Enable standby mode and click [Apply], the router will enter standby mode in 10 mins
10000029	请先删除已绑定的 MQTT 转发规则	Please delete the bound MQTT forwarding rules first
10000030	通道名称重复	Channel name already exists
10000031	检测到以下服务正在使用 WAN 接口，无法切换到 LAN 模式，请停用相关功能或修改接口分配后再重试	The following services are using the WAN interface and cannot be switched to LAN mode. Please disable the related functions or modify the interface allocation and try again
10010004	数量达到上限	Maximum number reached
10010005	参数错误	Invalid parameter

Error Code	Chinese Description	English Description
10010010	IP 地址非法	Invalid IP address
10010011	使用的端口冲突，请更换端口或检查设备配置	The port is in use, please change the port or check the device configuration
10019999	未知错误	Unknown error
10050001	请输入合法的密钥	Please enter a valid key

DeviceHub Activation Error Codes (302xxxxx)

Error Code	Chinese Description	English Description
30201001	激活设备，授权码错误	Activation failed, invalid Authentication Code
30201002	激活失败，账户名或密码错误	Activation failed, wrong account name or password
30201003	激活失败，DeviceHub 可允许连接设备数量已经达到最大值	Activation failed, your DeviceHub has reached the maximum number of allowed connections
30201004	激活失败，本台设备已被其他账户激活	Activation failed, the device has been activated by other account
30201005	激活失败，设备信息和设备管理服务器的信息不一致	Activation failed, device info is inconsistent with the info of device management server
30201006	激活失败，网络异常	Activation failed, network exception
30201007	激活失败，未知错误	Activation failed, unknown error

Email Service Error Codes (303xxxxx)

Error Code	Chinese Description	English Description
30301001	请检查邮箱账号密码是否正确	Authentication failed. Please double check your account info
30301002	请检查 SMTP 服务器配置是否正确	Please double check SMTP server settings
30301003	测试邮件已发出	The test email has been sent successfully
30301004	端口设置错误	Port setting error

DI Configuration Error Codes (404xxxxx)

Error Code	Chinese Description	English Description
40401001	请先配置 DI 并开启	Please configure DI and open it first

Advanced Options / Upgrade Error Codes (501xxxxx / 502xxxxx)

Error Code	Chinese Description	English Description
50101001	高级选项存在语法错误	Advanced option has a syntax error
50201001	请确认是否强制升级	Please confirm whether to force the upgrade
50201002	升级镜像不可用，设备将重启	The upgrade image is not available and the device will reboot

SDK / AppManager Error Codes (601xxxxx)

Error Code	Chinese Description	English Description
60101001	固件版本不匹配	The SDK version and firmware version do not match
60102001	密码修改成功，请重启 appmanager	The password was successfully modified. Please restart appmanager

DLMS Error Codes (1004xxxx)

Error Code	Chinese Description	English Description
10040001	物理设备已存在	Physical device already exists
10040002	obis_code 已存在	obis_code already exists
10040003	设备未连接	Device not connected
10040004	设备正在扫描	Device scanning in progress
10040005	串口状态错误	Serial port status error
10040006	对象已存在	Object already exists
10040007	对象不存在	Object does not exist
10040008	对象被使用	Object is in use