CoolMasterNet
CooLinkNet
CooLinkHub
CooLinkBridge

REST API Specification
# Table of Contents

1 **Configuration** .................................................. 3  

2 **REST API Specification** ................................. 4  
   2.1 Basics .......................................................... 4  
      Error States .................................................. 4  
   2.2 V1 API ........................................................ 5  
      Raw Command ............................................ 5  
   2.3 V2 API ........................................................ 6  
      ls, ls2 ......................................................... 6  

3 **Commands Reference** .................................... 8  
   3.1 rest ............................................................... 8
# 1 Configuration

CoolAutomation device must be properly configured to support REST API. Configuration is made via CoolAutomation's proprietary ASCII_IF protocol described in details in [Programmer Reference Manual (PRM)](http://example.com). REST Server can be enabled and configured with `rest` command. Below example shows how to enable REST Server.

```
> rest enable
OK, Boot Required!
```
## REST API Specification

### 2.1 Basics

**Media Type**
API relies on JSON to represent states of REST resources.

**Application Root**

/\API\_version\///\Device\_SN\///\Request\_str

- API version - Version code in format v\<X\>\.<Y\>
- Device\_SN - CoolAutomation device serial number
- Request\_str - Request string specific to API version

Examples:

/v1.0/device/283B96000049/raw?command=ls2
/v2.0/device/283B96002129/ls2

**Request-URI Encoding**
Request string may contain special characters that will be interpreted as shown below

<table>
<thead>
<tr>
<th>Special Character</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&amp;</code></td>
<td>- space</td>
</tr>
<tr>
<td><code>_</code></td>
<td>- underscore</td>
</tr>
<tr>
<td><code>.</code></td>
<td>- dot</td>
</tr>
</tbody>
</table>

For example request string "ls\&L1_100" will be interpreted as "ls L1.100".

### 2.1.1 Error States

To indicate errors related to REST Server the common HTTP Response Status Codes are used. Additional information regarding error reason is provided in response message-body with "error" JSON key. For example:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Content-Length: 24
{"error":"Wrong Device"}
```

<table>
<thead>
<tr>
<th>HTTP Response Status Code</th>
<th>&quot;error&quot; Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 Bad Request</td>
<td>Wrong v1 route</td>
<td>Wrong &lt;Request_str&gt; format</td>
</tr>
<tr>
<td></td>
<td>Wrong v2 route</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wrong API Version</td>
<td>&lt;API Version&gt; is not supported</td>
</tr>
<tr>
<td>403 Forbidden</td>
<td>Wrong Device</td>
<td>&lt;Device_SN&gt; does not match</td>
</tr>
<tr>
<td>404 Not Found</td>
<td>Wrong URI</td>
<td>URI format is wrong</td>
</tr>
<tr>
<td>405 Method Not Allowed</td>
<td>Wrong HTTP Method</td>
<td></td>
</tr>
<tr>
<td>413 Request Entity Too Large</td>
<td>Fragmented HTTP header</td>
<td>HTTP request is fragmented</td>
</tr>
<tr>
<td>501 Not Implemented</td>
<td>Unsupported V2.x command</td>
<td></td>
</tr>
</tbody>
</table>
2.2 V1 API

2.2.1 Raw Command

Request-Line

GET /v1.0/device/<Device_SN>/raw?command=<CMD> HTTP/1.1

Response message-body

```json
{
  "command": "<CMD_interpreted>",
  "data": [
    "<LINE1_optional>",
    "<LINE2_optional>",
    ...
    "<LINEN_optional>",
  ],
  "rc": "<Exit_Code>
}
```

This request is used to execute CoolAutomation device ASCII_IF command. CoolAutomation's proprietary ASCII_IF protocol is described in details in [Programmer Reference Manual (PRM)](https://www.coolautomation.com).

Examples

- **Turn ON indoor unit L1.100**

  Request: /v1.0/device/283B96002128/raw?command=on&L1_100

  Response:

  ```json
  {
    "command": "on L1.100",
    "data": [],
    "rc": "OK"
  }
  ```

- **Query status of HVAC lines with ASCII_IF line command**

  Request: /v1.0/device/283B96002128/raw?command=line

  Response:

  ```json
  {
    "command": "line",
    "data": [
      " L1: DK Master U00/G06 myid:0B",
      " Tx:749/749 Rx:749/749 TO:0/0 CS:0/0 Col:0/0 NAK:0/0",
      " L2: Unused ",
      " Tx:0/0 Rx:0/0 TO:0/0 CS:0/0 Col:0/0 NAK:0/0",
      " L3: CG5 Modbus Address:0x50(80) 9600_8N1 ",
      " Tx:0/0 Rx:0/0 TO:0/0 CS:0/0 Col:0/0 NAK:0/0",
      " L4: Unused ",
      " Tx:0/0 Rx:0/0 TO:0/0 CS:0/0 Col:0/0 NAK:0/0",
    ],
  }
  ```
" L5: Unused ",
" Tx:0/0 Rx:0/0 TO:0/0 CS:0/0 Col:0/0 NAK:0/0"
" L6: Unused ",
" Tx:0/0 Rx:0/0 TO:0/0 CS:0/0 Col:0/0 NAK:0/0"
" L7: Unused ",
" Tx:0/0 Rx:0/0 TO:0/0 CS:0/0 Col:0/0 NAK:0/0"
" L8: Unused ",
" Tx:0/0 Rx:0/0 TO:0/0 CS:0/0 Col:0/0 NAK:0/0"

],
"rc": "OK"
}

2.3 V2 API

2.3.1 ls, ls2

Request-Line
GET /v2.0/device/<Device_SN>/ls<&UID_optional> HTTP/1.1
or
GET /v2.0/device/<Device_SN>/ls2<&UID_optional> HTTP/1.1

Response message-body
{  
  "command": "<CMD_interpreted>",
  "data": [
    {<JSON1_optional>},
    {<JSON2_optional>},
    ...
    {<JSONn_optional>}
  ],
  "rc": "<Exit_Code>"
}

This request is used to execute ls or ls2 command with optional UID. Commands provide indoor unit(s) status. If UID is omitted all indoor units connected to CoolAutomation device will be listed. Response is fully JSON formatted. Response JSON keys have following meaning:

<table>
<thead>
<tr>
<th>JSON Key</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;uid&quot;</td>
<td>Indoor unit UID</td>
</tr>
<tr>
<td>&quot;onoff&quot;</td>
<td>ON/OFF status</td>
</tr>
<tr>
<td>&quot;st&quot;</td>
<td>Set Temperature</td>
</tr>
<tr>
<td>&quot;rt&quot;</td>
<td>Room Temperature</td>
</tr>
<tr>
<td>&quot;fspeed&quot;</td>
<td>Fan Speed</td>
</tr>
<tr>
<td>&quot;mode&quot;</td>
<td>Operation Mode</td>
</tr>
<tr>
<td>&quot;fls&quot;</td>
<td>Failure Code</td>
</tr>
<tr>
<td>&quot;filt&quot;</td>
<td>Filter Cleaning</td>
</tr>
<tr>
<td>&quot;dmnd&quot;</td>
<td>Demand (Therm_ON)</td>
</tr>
</tbody>
</table>

Example
Request: GET /v2.0/device/283B96002128/ls2&L1_100
Response:
{
  "command": "ls2 L1.100",
  "data": [
    {
      "uid": "L1.100",
      "onoff": "ON",
      "st": "062.4F",
      "rt": "076.1F",
      "fspeed": "Low",
      "mode": "Auto",
      "flr": "OK",
      "filt": "#",
      "dmnd": "0"
    }
  ],
  "rc": "OK"
}
3 Commands Reference

rest

3.1 rest

SYNOPSIS
rest
  rest enable
  rest disable
  rest port <PORT>

DESCRIPTION
- Without parameters rest command displays current REST Server configuration.
- rest port command is used to configure TCP/IP listen port used by REST Server. Default port number is 10103.

EXAMPLE

```
>rest
REST      : enabled
Listen port : 10103

>rest port 8080
OK, Boot Required!
```