REST API

Overview

The term "REST API" generally refers to an API accessed via HTTP protocol at a predefined set of resource-oriented URLs.
Starting from **RouterOS v7.1beta4**, it is implemented as a JSON wrapper interface of the console API. It allows to create, read, update and delete resources and call arbitrary console commands.

To start to use REST API, the `www-ssl` service [link to IP services] must be configured and running. When the SSL service is enabled the REST service can be accessed by connecting to `https://<routers_IP>/rest`.

The easiest way to start is to use cURL, wget, or any other HTTP client even RouterOS fetch tool.

```
$ curl -k -u admin: https://10.155.101.214/rest/system/resource
[{"architecture-name":"tile","board-name":"CCR1016-12S-1S+",
"build-time":"Dec/04/2020 14:19:51","cpu":"tilegx","cpu-count":"16",
"cpu-frequency":"1200","cpu-load":"1","free-hdd-space":"83439616",
"free-memory":"1503133696","platform":"MikroTik",
"total-hdd-space":"134217728","total-memory":"2046820352",
"uptime":"2d20h12m20s","version":"7.1beta4 (development)"}]
```

Authentication

Authentication to the REST API is performed via HTTP Basic Auth. Provide your Username and password are the same as for the console user (by default "admin" with no password).

> ⚠ You have to set up certificates to use secure HTTPS, if self-signed certs are used, then CA must be imported to the trusted root. However, for testing purposes, it is possible to connect insecurely (for cUrl use -k flag, for wget use --no-check-certificate)

JSON format

Server broadly follows ECMA-404 standard, with following notes:

- In JSON replies all object values are encoded as strings, even if the underlying data is a number or a boolean.
- The server also accepts numbers in octal format (begins with 0) and hexadecimal format (begins with 0x). If the numbers are sent in a string format, they are assumed to be in decimal format.
- Numbers with exponents are not supported.

HTTP Methods

Below is a table summarising supported HTTP methods

HTTP Verv	CRUD	ROS	Description
GET	Read	print	To get the records.
PATCH	Update/Modify	set	To update a single record.
PUT	Create	add	To create a new record.
DELETE	Delete	remove	To delete a single record.
POST			Universal method to get access to all console commands.

URL Format

GET

This method allows getting the list of all records or a single record from the specified menu encoded in the URL.
For example, get all IP addresses (equivalent to the '`ip/address/print`' command from the CLI):

```
$ curl -k -u admin: https://10.155.101.214/rest/ip/address
[{".id":"*1","actual-interface":"ether2","address":"10.0.0.111/24","disabled":"false",
"dynamic":"false","interface":"ether2","invalid":"false","network":"10.0.0.0"},
{".id":"*2","actual-interface":"ether3","address":"10.0.0.109/24","disabled":"true",
"dynamic":"false","interface":"ether3","invalid":"false","network":"10.0.0.0"}]
```

To return a single record, append the ID at the end of the URL:

```
$ curl -k -u admin: https://10.155.101.214/rest/ip/address/*1
{".id":"*1","actual-interface":"ether2","address":"10.0.0.111/24","disabled":"false",
"dynamic":"false","interface":"ether2","invalid":"false","network":"10.0.0.0"}
```

If table contains named parameters, then name instread of ID can be used, for example, get ether1:

```
$ curl -k -u admin: https://10.155.101.214/rest/interface/ether1
```

It is also possible to filter the output, for example, return only valid addresses that belong to the 10.155.101.0 network:

```
$ curl -k -u admin: "https://10.155.101.214/rest/ip/address?network=10.155.101.0&dynamic=true"
[{".id":"*8","actual-interface":"sfp12","address":"10.155.101.214/24","disabled":"false",
"dynamic":"true","interface":"sfp12","invalid":"false","network":"10.155.101.0"}]
```

Another example returns only addresses on the "dummy" interface and with the comment "test":

```
$ curl -k -u admin: 'https://10.155.101.214/rest/ip/address?comment=test&interface=dummy'
[{".id":"*3","actual-interface":"dummy","address":"192.168.99.2/24","comment":"test",
"disabled":"false","dynamic":"false","interface":"dummy","invalid":"false","network":"192.168.99.0"}]
```

If you want to return only specific properties, you can use the '`.proplist`', followed by the '=' and a list of comma-separated properties. For example, to show only the address and if it's disabled:

```
$ curl -k -u admin: https://10.155.101.214/rest/ip/address?.proplist=address,disabled
[{"address":"10.0.0.111/24","disabled":"false"},{"address":"10.0.0.109/24","disabled":"true"}]
```

PATCH

This method is used to update a single record. Set PATCH call body as a JSON object which contains fields and values of the properties to be updated. For example, add a comment:

```
$ curl -k -u admin: -X PATCH https://10.155.101.214/rest/ip/address/*3 \
  --data '{"comment": "test"}' -H "content-type: application/json"
{".id":"*3","actual-interface":"dummy","address":"192.168.99.2/24","comment":"test",
"disabled":"false","dynamic":"false","interface":"dummy","invalid":"false","network":"192.168.99.0"}
```

In case of a successful update, the server returns the updated object with all its parameters.

PUT

A method is used to create new records in the menu encoded in the URL. The body should be set as a JSON object containing parameters applied to the newly created record.

In case of success, the server returns the created object with all its parameters.

Only one resource can be created in a single request.

For example, add an IP address to a dummy interface:

```
$ curl -k -u admin: -X PUT https://10.155.101.214/rest/ip/address \
  --data '{"address": "192.168.111.111", "interface": "dummy"}' -H "content-type: application/json"
{".id":"*A","actual-interface":"dummy","address":"192.168.111.111/32","disabled":"false",
"dynamic":"false","interface":"dummy","invalid":"false","network":"192.168.111.111"}
```

DELETE

This method is used to delete the record with a specified ID from the menu encoded in the URL. If the deletion has been succeeded, the server responds with an empty response. For example, call to delete the record twice, on second call router will return 404 error:

```
$ curl -k -u admin: -X DELETE https://10.155.101.214/rest/ip/address/*9
$ curl -k -u admin: -X DELETE https://10.155.101.214/rest/ip/address/*9
{"error":404,"message":"Not Found"}
```

POST

All the API features are available through the `POST` method. The command word is encoded in the header and optional parameters are passed in the JSON object with the corresponding fields and values. For example, to change the password of the active user, send

```
POST https://router/rest/password
{"old-password":"old","new-password":"N3w", "confirm-new-password":"N3w"}
```

REST response is structured similar to API response:

- If the response contains '`!re`' sentences (records), the JSON reply will contain a list of objects.
- If the '`!done`' sentence contains data, the JSON reply will contain an object with the data.
- If there are no records or data in the '`!done`' sentence, the response will hold an empty list.

There are two special keys: `.proplist` and `.query`, which are used with the `print` command word. Read more about APIs responses, prop lists, and queries in the API documentation.

Proplist

The '`.proplist`' key is used to create `.proplist` attribute word. The values can be a single string with comma-separated values:

```
POST https://router/rest/interface/print
{".proplist":"name,type"}
```

or a list of strings:

```
POST https://router/rest/interface/print
{".proplist":["name","type"]}
```

For example, return address and interface properties from the ip/address list:

```
$ curl -k -u admin: -X POST https://10.155.101.214/rest/ip/address/print\
  --data '{"_proplist": ["address","interface"]}' -H "content-type: application/json"
[{"address":"192.168.99.2/24","interface":"dummy"},
{"address":"172.16.5.1/24","interface":"sfpplus1"},
{"address":"172.16.6.1/24","interface":"sfp2"},
{"address":"172.16.7.1/24","interface":"sfp3"},
{"address":"10.155.101.214/24","interface":"sfp12"},
{"address":"192.168.111.111/32","interface":"dummy"}]
```

Query

The '`.query`' key is used to create a query stack. The value is a list of query words. For example this POST request :

```
POST https://router/rest/interface/print
{".query":["type=ether","type=vlan","#|!"]}
```

is equivalent to this API sentence

```
/interface/print
?type=ether
?type=vlan
?#|!
```

For example, let's combine '*query*' and '*proplist*', to return '.id', 'address', and 'interface' properties for all dynamic records and records with the network 192.168.111.111

```
$ curl -k -u admin: -X POST https://10.155.101.214/rest/ip/address/print \
  --data '{".proplist": [".id","address","interface"], ".query": ["network=192.168.111.111","dynamic=true","
#|"]}'\
  -H "content-type: application/json"
[{".id":"*8","address":"10.155.101.214/24","interface":"sfp12"},
{".id":"*A","address":"192.168.111.111/32","interface":"dummy"}]
```

Timeout

If the command runs indefinitely, it will timeout and the connection will be closed with an error. The current timeout interval is 60 seconds. To avoid timeout errors, add a parameter that would sufficiently limit the command execution time.

⚠ Timeout is not affected by the parameters passed to the commands. If the command is set to run for an hour, it will terminate early and return an error message.

For example, let's see what we get when the ping command exceeds the timeout and how to prevent this by adding a count parameter:

```
$ curl -k -u admin: -X POST https://10.155.101.214/rest/ping \
  --data '{"address":"10.155.101.1"}' \
  -H "content-type: application/json"
{"detail":"Session closed","error":400,"message":"Bad Request"}

$ curl -k -u admin: -X POST https://10.155.101.214/rest/ping \
  --data '{"address":"10.155.101.1","count":"4"}' \
  -H "content-type: application/json"
[{"avg-rtt":"453us","host":"10.155.101.1","max-rtt":"453us","min-rtt":"453us","packet-loss":"0","received":"1","
sent":"1","seq":"0","size":"56","time":"453us","ttl":"64"},
{"avg-rtt":"417us","host":"10.155.101.1","max-rtt":"453us","min-rtt":"382us","packet-loss":"0","received":"2","
sent":"2","seq":"1","size":"56","time":"382us","ttl":"64"},
{"avg-rtt":"495us","host":"10.155.101.1","max-rtt":"650us","min-rtt":"382us","packet-loss":"0","received":"3","
sent":"3","seq":"2","size":"56","time":"650us","ttl":"64"},
{"avg-rtt":"461us","host":"10.155.101.1","max-rtt":"650us","min-rtt":"359us","packet-loss":"0","received":"4","
sent":"4","seq":"3","size":"56","time":"359us","ttl":"64"}]
```

Another example is a bandwidth test tool, which can be limited by providing run duration:

```
$ curl -k -u admin: -X POST 'https://10.155.101.214/rest/tool/bandwidth-test' \
  --data '{"address":"10.155.101.1","duration":"2s"}' \
  -H "content-type: application/json"
[{".section":"0","connection-count":"20","direction":"receive","lost-packets":"0",
"random-data":"false","rx-10-second-average":"0","rx-current":"0","rx-size":"1500",
"rx-total-average":"0",
"status":"connecting"},
{".section":"1","connection-count":"20","direction":"receive","duration":"1s",
"lost-packets":"0","random-data":"false","rx-10-second-average":"0","rx-current":"0",
"rx-size":"1500","rx-total-average":"0",
"status":"running"},
{".section":"2","connection-count":"20","direction":"receive","duration":"2s",
"lost-packets":"581175","random-data":"false","rx-10-second-average":"854372352",
"rx-current":"854372352","rx-size":"1500","rx-total-average":"854372352",
"status":"running"},
{".section":"3","connection-count":"20","direction":"receive","duration":"3s",
"lost-packets":"9014","random-data":"false","rx-10-second-average":"891979008",
"rx-current":"929585664","rx-size":"1500","rx-total-average":"891979008",
"status":"done testing"}]
```

Errors

The success or failure of the API calls is indicated in the HTTP status code. In case of failure (status code 400 or larger), the body of the response contains a JSON object with the error code, a description of the error, and optional error details. For example, trying to delete an interface will return

```
{"error":406,"message":"Not Acceptable","detail":"no such command or directory (remove)"}
```