Fiberstore

Configuration Guide
Contents

1 Configuring Service.................................................................3
  1.1 Overview..............................................................................3
  1.2 Configuring RPC API service.............................................3
  1.3 Configuring RPC API service with HTTP Authentication..........4
  1.4 Validation Commands.......................................................4

2 RPC API Spec........................................................................6
  2.1 Overview..............................................................................6
  2.2 JSON-RPC Request............................................................6
  2.3 JSON-RPC Response.........................................................7
  2.4 Python Client Example Code.............................................7
  2.5 JSON-RPC Error Code......................................................8
  2.6 RPC-API Error Code.......................................................8
1 Configuring Service

1.1 Overview
RPC API service allows user to configure and monitor the Fiberstore switch system through Remote Procedure Calls (RPC) from your program.

The service currently supports JSON-RPC over HTTP protocol together with HTTP Basic authentication.

1.2 Configuring RPC API service
User could enable the RPC API service by the following steps:

**Switch1**

<table>
<thead>
<tr>
<th>Switch# configure terminal</th>
<th>Enable global configuration mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch(config)# service rpc-api enable port 80</td>
<td>Enable RPC API service using TCP port 80 (HTTP). By default, the RPC-API service listens on TCP port 80 (HTTP)</td>
</tr>
<tr>
<td>Switch(config)# exit</td>
<td>Exit the configuration mode</td>
</tr>
</tbody>
</table>

User could disable the RPC API service by the following steps:

**Switch1**

<table>
<thead>
<tr>
<th>Switch# configure terminal</th>
<th>Enable global configuration mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch(config)# service rpc-api disable</td>
<td>Disable RPC API service</td>
</tr>
<tr>
<td>Switch(config)# exit</td>
<td>Exit the configuration mode</td>
</tr>
</tbody>
</table>
1.3 Configuring RPC API service with HTTP Authentication

User could configure the HTTP authentication mode of RPC API service.

Currently, only HTTP Basic authentication is supported. User will receive status code: 401 (Unauthorized access) if user provides invalid user name or password.

**Switch1**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch# configure terminal</td>
<td>Enable global configuration mode</td>
</tr>
<tr>
<td>Switch(config)# username Fiberstore password Fiberstore</td>
<td>Configure user name (Fiberstore) and password (Fiberstore), for HTTP authentication.</td>
</tr>
<tr>
<td>Switch(config)# service rpc-api auth-mode basic</td>
<td>Enable HTTP Basic authentication</td>
</tr>
<tr>
<td>Switch(config)# exit</td>
<td>Exit the configuration mode</td>
</tr>
</tbody>
</table>

User could disable HTTP authentication by the following steps:

**Switch1**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch# configure terminal</td>
<td>Enable global configuration mode</td>
</tr>
<tr>
<td>Switch(config)# no service rpc-api auth-mode basic</td>
<td>Disable HTTP authentication</td>
</tr>
<tr>
<td>Switch(config)# exit</td>
<td>Exit the configuration mode</td>
</tr>
</tbody>
</table>

**NOTE**

HTTP authentication settings of RPC API service will take effect after you restart this service or reboot the system.

1.4 Validation Commands

All current RPC API service configurations can be displayed.
Switch1

<table>
<thead>
<tr>
<th>Switch# show services rpc-api</th>
<th>Show the current RPC API service configurations</th>
</tr>
</thead>
</table>

RPC API services configuration:
HTTP server: running, port: 80, authentication mode: none
2.1 Overview

Fiberstore RPC API service uses standard JSON-RPC over HTTP protocol to communicate the switch and your program. User may issue switch CLI commands through JSON-RPC method: ‘executeCmds’. By default, the CLI mode is in privileged EXEC mode (#).

User could send JSON-RPC request via an HTTP POST request to URL:

http://<switch_management_ip_address>:<switch_tcp_port_number>/command-api.

The detailed JSON-RPC request and response are show below:

2.2 JSON-RPC Request

```json
{
  "params": [
    {
      "format":"text",
      "version":1,
      "cmds":[
        "show run",
        "config t",
        "vlan database",
        "vlan 1-8",
        "interface eth-0-1",
        "switchport mode trunk",
        "switchport trunk allowed vlan add 2",
        "shutdown",
        "end",
        "show interface switchport"
      ]
    }
  ]
}
```
JSON RPC protocol version. Always 2.0.

Method to run the switch CLI commands

JSON RPC unique identifier

JSON-RPC protocol version. Always 2.0.

JSON RPC unique identifier

Result list of objects from each CLI command executed.

Output information of CLI Command 1.
The Original ASCII output information returned from CLI command if this command is successfully executed.

Error code if it is available.

Error description if it is available.

Warnings if it is available.

Formatted JSON object will also be returned if it is available.

Output information of CLI Command 2.

Output information of CLI Command 3.

Output information of CLI Command 4.

Output information of CLI Command 5.

Output information of CLI Command 6.

Output information of CLI Command 7.

Output information of CLI Command 8.

Output information of CLI Command 9.

Output information of CLI Command 10.

Python Client Example Code

Here is an example code using ‘pyjsonrpc’ library:
import pyjsonrpc
import json

http_client = pyjsonrpc.HttpClient(
    url = "http://10.10.39.64:80/command-api",
    username = "Fiberstore",
    password = "Fiberstore"
)

cmds = {}
cmd_list = ["show run", "config t", "vlan database", "vlan 1-8", "interface eth-0-1", "switchport mode trunk", "switchport trunk allowed vlan add 2", "shutdown", "end", "show interface switchport"]

cmds["cmds"] = cmd_list
cmds["format"] = 'text'
cmds["version"] = 1

try:
    response = http_client.call("executeCmds", cmds)
    print("json response:");
    json_result = json.dumps(response, indent=4)
    print(json_result)
except Exception, e:
    if e.code == 401:
        print "Unauthorized user"
    else:
        print e.message
        print e.data

2.5 JSON-RPC Error Code
Here is a list of JSON-RPC 2.0 error code:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-32700</td>
<td>Parse error</td>
</tr>
<tr>
<td>-32600</td>
<td>Invalid Request</td>
</tr>
<tr>
<td>-32601</td>
<td>Method not found</td>
</tr>
<tr>
<td>-32602</td>
<td>Invalid param</td>
</tr>
<tr>
<td>-32603</td>
<td>Internal error</td>
</tr>
</tbody>
</table>

2.6 RPC-API Error Code
Here is a list of RPC-API error code:
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1000</td>
<td>General error</td>
</tr>
<tr>
<td>-2001</td>
<td>JSON RPC API Error: unsupported API version</td>
</tr>
<tr>
<td>-2002</td>
<td>JSON RPC API Error: must specify 'params' with 'cmds' in JSON RPC</td>
</tr>
<tr>
<td>-2003</td>
<td>JSON RPC API Error: unsupported command response format</td>
</tr>
<tr>
<td>-3001</td>
<td>Command execution failed: timed out</td>
</tr>
<tr>
<td>-3002</td>
<td>Command execution failed: unsupported command</td>
</tr>
<tr>
<td>-3003</td>
<td>Command execution failed: unauthorized command</td>
</tr>
<tr>
<td>-3004</td>
<td>Command execution failed: the string does not match any command in current mode</td>
</tr>
<tr>
<td>-3005</td>
<td>Command execution failed: can't convert to JSON format</td>
</tr>
<tr>
<td>-3006</td>
<td>Command execution failed: command list too short</td>
</tr>
<tr>
<td>-3007</td>
<td>Command execution failed: command list too long</td>
</tr>
</tbody>
</table>