FS.COM

FSOS

RPC API Configuration Guide

Contents

1 Configuring Service

1.1 Overview

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

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

Note: this feature is only available on E350-SDN image.

1.2 Configuring RPC API service

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

Switch1

Switch# configure terminal	Enable global configuration mode
Switch(config)# service rpc-api enable port 80	Enable RPC API service using TCP port 80 (HTTP). By default, the RPC-API service listens on TCP port 80 (HTTP)
Switch(config)# exit	Exit the configuration mode

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

Switch1

Switch# configure terminal	Enable global configuration mode
Switch(config)# service rpc-api disable	Disable RPC API service

Switch(config)# exit	Exit the configuration mode

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

Switch# configure terminal	Enable global configuration mode
Switch(config)# username FS password FS	Configure user name (FS) and password (FS), for HTTP authentication.
Switch(config)# service rpc-api auth-mode basic	Enable HTTP Basic authentication
Switch(config)# exit	Exit the configuration mode

User could disable HTTP authentication by the following steps:

Switch1

Switch# configure terminal	Enable global configuration mode
Switch(config)# no service rpc-api auth-mode	Disable HTTP authentication
Switch(config)# exit	Exit the configuration mode

📖 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

Switch# show services rpc-api	Show the current RPC API service configurations

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

2 RPC API Spec

2.1 Overview

FS RPC API service uses standard JSON-RPC over HTTP protocol to communicat the switch and your program. User may issue switch CLI commands through JSON-RPC method: 'executeCmds'. By default, the CLI mode is in priviledged 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

```
{
  "params":[                          Parameters for command
    {
      "format":"text",                Expected response format, can be 'text' or 'json',
                                      the default format is 'text'
      "version":1,                    The API version
      "cmds":[                        List of CLI commands
        "show run",                   CLI command 1
        "config t",                   CLI command 2
        "vlan database",              CLI command 3
        "vlan 1-8",                   CLI command 4
        "interface eth-0-1",          CLI command 5
        "switchport mode trunk",      CLI command 6
        "switchport trunk allowed vlan add 2",    CLI command 7
        "shutdown",                   CLI command 8
        "end",                        CLI command 9
        "show interface switchport"   CLI command 10
      ]
    }
```

```
],
"jsonrpc":"2.0",                                    JSON RPC protocol version. Always 2.0.
"method":"executeCmds",                             Method to run the switch CLI commands
"id":"70853aff-af77-420e-8f3c-fa9430733a19"         JSON RPC unique identifier
}
```

2.3 JSON-RPC Response

```
{
  "jsonrpc":"2.0",                                  JSON RPC protocol version. Always 2.0.
  "id":"70853aff-af77-420e-8f3c-fa9430733a19",      JSON RPC unique identifier
  "result":[                                         Result list of objects from each CLI command
                                                    executed.

    {
      "sourceDetails":"version 5.1.6.fcs\n!\n …",    Output information of CLI Command 1.
                                                    The Original ASCII output information
                                                    returned from CLI command if this command
                                                    is successfully executed.
      "errorCode":-1003,                            Error code if it is available.
      "errorDesc":"unsupported command…",           Error description if it is available.
      "warnings":"% Invalid…",                       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.
    {
      "sourceDetails":" Interface name        : eth-0-1\n Switchport mode      : trunk\n …\n"
    }                                                Output information of CLI Command 10.
  ]
}
```

2.4 Python Client Example Code

Here is an example code using 'pyjsonrpc' library:

```python
import pyjsonrpc
import json

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

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:

Error Code	Description
-32700	Parse error
-32600	Invalid Request
-32601	Method not found
-32602	Invalid param
-32603	Internal error

2.6 RPC-API Error Code

Here is a list of RPC-API error code:

Error Code	Description
-1000	General error
-2001	JSON RPC API Error: unsupported API version
-2002	JSON RPC API Error: must specify 'params' with 'cmds' in JSON RPC
-2003	JSON RPC API Error: unsupported command response format
-3001	Command execution failed: timed out
-3002	Command execution failed: unsupported command
-3003	Command execution failed: unauthorized command
-3004	Command execution failed: the string does not match any command in current mode
-3005	Command execution failed: can't convert to JSON format
-3006	Command execution failed: command list too short
-3007	Command execution failed: command list too long