Extreme®
Connect Beyond the Network

ExtremeXOS Machine to Machine Interface (MMI) Application

Release Notes v1.0.0

121152-00

Published February 2016

Table of Contents

Preface

Text Conventions

The following tables list text conventions that are used throughout this guide.

Table 1: Notice Icons

Icon	Notice Type	Alerts you to...
👉	General Notice	Helpful tips and notices for using the product.
📝	Note	Important features or instructions.
⚠	Caution	Risk of personal injury, system damage, or loss of data.
⚡	Warning	Risk of severe personal injury.
🆕	New	This command or section is new for this release.

Table 2: Text Conventions

Convention	Description
`Screen displays`	This typeface indicates command syntax, or represents information as it appears on the screen.
The words enter and type	When you see the word "enter" in this guide, you must type something, and then press the Return or Enter key. Do not press the Return or Enter key when an instruction simply says "type."
[Key] names	Key names are written with brackets, such as **[Return]** or **[Esc]**. If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press **[Ctrl]**+**[Alt]**+**[Del]**
Words in italicized type	Italics emphasize a point or denote new terms at the place where they are defined in the text. Italics are also used when referring to publication titles.

Platform-Dependent Conventions

Unless otherwise noted, all information applies to all platforms supported by ExtremeXOS software, which are the following:

- ExtremeSwitching® switches
- Summit® switches
- SummitStack™

When a feature or feature implementation applies to specific platforms, the specific platform is noted in the heading for the section describing that implementation in the *ExtremeXOS Command Reference Guide*. In many cases, although the command is available on all platforms, each platform uses specific keywords. These keywords specific to each platform are shown in the Syntax Description and discussed in the Usage Guidelines.

Terminology

When features, functionality, or operation is specific to a switch family, such as ExtremeSwitching or Summit, the family name is used. Explanations about features and operations that are the same across all product families simply refer to the product as the *switch*.

Providing Feedback to Us

We are always striving to improve our documentation and help you work better, so we want to hear from you! We welcome all feedback but especially want to know about:

- Content errors or confusing or conflicting information.
- Ideas for improvements to our documentation so you can find the information you need faster.
- Broken links or usability issues.

If you would like to provide feedback to the Extreme Networks Information Development team about this document, please contact us using our short online feedback form. You can also email us directly at internalinfodev@extremenetworks.com.

Getting Help

If you require assistance, you can contact Extreme Networks using one of the following methods:

- Global Technical Assistance Center (GTAC) for Immediate Support
    - Phone: 1-800-872-8440 (toll-free in U.S. and Canada) or 1-603-952-5000. For the Extreme Networks support phone number in your country, visit: www.extremenetworks.com/support/contact
    - Email: support@extremenetworks.com. To expedite your message, enter the product name or model number in the subject line.
- GTAC Knowledge — Get on-demand and tested resolutions from the GTAC Knowledgebase, or create a help case if you need more guidance.
- The Hub — A forum for Extreme customers to connect with one another, get questions answered, share ideas and feedback, and get problems solved. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.
- Support Portal — Manage cases, downloads, service contracts, product licensing, and training and certifications.

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Network products
- A description of the failure
- A description of any action(s) already taken to resolve the problem

- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related Return Material Authorization (RMA) numbers

Extreme Networks Publications

General

Product documentation is available at: http://documentation.extremenetworks.com. Release notes are available at: www.extremenetworks.com/support/release-notes

Open Source Declarations

Some software files have been licensed under certain open source licenses. More information is available at: www.extremenetworks.com/support/policies/software-licensing

1 Introduction

Introduction

The ExtremeXOS Machine to Machine Interface (MMI) application is intended to provide a simple interface for automated remote management. MMI is implemented as a Remote Procedure Call (RPC) using JSON (RFC 4627) data format to call EXOS CLI and pass Python scripting inline.

MMI is compatible with ExtremeXOS 21.1 and later.

Configuration

To initiate the MMI (json) remote procedure call: `http://<ipAddress>/jsonrpc/`. Where *`<ipAddress>`* is the IP address of the switch where execution takes place.

Incorrect or Missing Username/Password

MMI uses the same authentication mechanism as the CLI login. "Admin" level login permission is required. When an invalid username or password is provided, or the username/password is missing from the JSONRPC request, the following response is returned from the switch.

Response

```
{
"error": {
"code": 401,
"data": null,
"message": "InvalidCredentialsError: 401 UNAUTHORIZED",
"name": "InvalidCredentialsError"
},
"id": "1",
"jsonrpc": "2.0"
}
```

2 Using MMI (JSON)

The HTTP POST method is used for all JSONRPC transactions.

Request

The JSONRPC data format is specified in (http://www.jsonrpc.org/specification), and repeated here to illustrate how they are used in ExtremeXOS.

jsonrpc	A String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0".
id	An identifier established by the Client that MUST contain a String, Number, or NULL value if included. If it is not included it is assumed to be a notification and an EXOS response will not be returned.
method	ExtremeXOS supports 2 methods: • CLI (see CLI Method on page 8) • Python (see Python Method on page 14)
params	• For the CLI method, params contains a string of ‘;’ separated list of ExtremeXOS CLI commands. More that one command may be passed in a single JSONRPC transaction by terminating the CLI command with a ‘;’ before starting the next. (For example, "params":["create vlan 10-20;show vlan"]. In this example there are two CLI commands separated by ‘;’. • For the Python method, params contains the entire Python script to be run on the switch. (For example, "params":["print\nprint 'Hello World'\nprint 'How are ya?'"]

Response

jsonrpc	A String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0".
id	An identifier established by the Client that MUST contain a string, number, or null value, if included. If it is not included, it is assumed to be a notification and an ExtremeXOS response is not returned.
result	Contains the ExtremeXOS reply to the JSONRPC request. See below for the different content depending on a CLI or Python method.
error	If there was an error processing the request, the error field contains a message indicating what went wrong.

CLI Method

For the CLI method, there are two different output formats depending on the CLI command:

- `debug cfgmgr show` …' commands (see debug cfgmgr show Commands on page 11)
- All other ExtremeXOS CLI commands

For ExtremeXOS CLI commands that create or update a switch, the response contains an empty "CLIoutput" for each CLI command processed, just like using a CLI shell where a successful command simply returns the prompt.

Example

Request

```
{"method":"cli","id":"10","jsonrpc":"2.0","params":["create vlan myvlan tag
101;config vlan myvlan add ports 12 tagged"]}
```

Response

```
{
  "jsonrpc": "2.0",
  "id": "10",
  "result": [
    [
      {
        "CLIoutput": ""
      }
    ],
    [
      {
        "CLIoutput": ""
      }
    ]
  ]
}
```

Failed Command

The following is an example of a failed CLI command. In this example, the same command from above is sent which tries to create a VLAN with the same name. The error field contains a message of the failed command.

Request

```
{"method":"cli","id":"10","jsonrpc":"2.0","params":["create vlan myvlan tag
101;config vlan myvlan add ports 12 tagged"]}
```

Response

```
{
  "jsonrpc": "2.0",
  "id": "10",
  "result": [
    {
      "error": "error processing command create vlan myvlan tag 101"
    },
    [
      {
        "CLIoutput": ""
      }
    ]
  ]
}
```

Show Commands

For ExtremeXOS show CLI commands, the response contains a series of key value pairs:

- CLIoutput—this field contains the formatted CLI as it would appear on an ExtremeXOS shell session, such as a console or Telnet session. It is useful if the remote JSONRPC caller wishes to manipulate or inspect the actual CLI display programmatically.
- A series of JSON encode data structures. For ExtremeXOS show commands, the data structures used in creating the CLI output.

Example

The following example uses the ExtremeXOS CLI `show ports 1 info` command:

Request

```
{"method":"cli","id":"10","jsonrpc":"2.0","params":["show ports 1 info"]}
```

Response

```
{
  "jsonrpc": "2.0",
  "id": "10",
  "result": [
    {
      "CLIoutput": "Port      Flags               Link      ELSM Link Num
Num   Num Jumbo QOS     Load\n                             State     /OAM
UPS STP VLAN Proto  Size profile Master
\n==============================================================
=========\n1         Em------e--fMB---x- ready     - / -   0   1    1     1
9216 none
\n==============================================================
=========\n> indicates Port Display Name truncated past 8 characters\nFlags :
a - Load Sharing Algorithm address-based,\n        b - Rx and Tx Flow Control
Enabled, B - Broadcast Flooding Enabled,\n        D - Port Disabled, e -
Extreme Discovery Protocol Enabled,\n        E - Port Enabled, f - Unicast
Flooding Enabled,\n        F - Priority Flow Control Enabled, G - MLAG
Enabled, i - Isolation,\n        j - Jumbo Frame Enabled, l - Load Sharing
Enabled,\n        L - Extreme Link Status Monitoring Enabled,\n        m -
MACLearning Enabled, M - Multicast Flooding Enabled,\n        n - Ingress TOS
Enabled, o - Dot1p Replacement Enabled,\n        O - Ethernet OAM Enabled, p
- Load Sharing Algorithm port-based,\n        P - Software redundant
port(Primary),\n        R - Software redundant port(Redundant), s - diffserv
Replacement Enabled,\n        v - Vman Enabled, w - MACLearning Disabled with
Forwarding,\n        x - Rx Flow Control Enabled\n"
```

```
    },
    { These data structures were used to create the information in the CLI
output
      "status": "SUCCESS",
      "show_ports_info": {
        "qosProfile": 0,
        "lsEnabled": 0,
        "noProtocols": 1,
        "ethOamLinkState": 2,
        "elsmLinkState": 2,
        "portList": 1,
        "noVlansOnPort": 1,
        "linkState": 0,
        "jumboSize": 9216,
        "flags": "Em------e--fMB---x-",
        "noLinkUpChanges": 0,
        "diagTestStatus": 1,
        "port": 1,
        "driveStrength": 0
      }
    },
    {
      "status": "SUCCESS",
      "stp_port_info": {
        "port_count": 1
      }
    }
  ]
}
```

debug cfgmgr show Commands

For **`debug cfgmgr show`** CLI commands, the output is formatted as described below. This debug command allows direct access to the CM data objects by formatting the CLI request command in the manner that the back-end CM software is expecting. This is useful when retrieving multiple instances for a single CM object, but, in most cases, requires knowledge of the CM interface for that object.

Retrieving System Information from dm.dm_system Object

Request

```
{"method":"cli","id":"10","jsonrpc":"2.0","params":["debug cfgmgr show one
dm.dm_system"]}
```

Response

```
{
  "jsonrpc": "2.0",
  "id": "10",
  "result": {
    "data": [
      {
        "disable_temp_check": null,
        "sysUpTime": "7444600",
        "power_off_on_failure": null,
        "reboot_loop_period": "0",
        "message": "",
        "rate_limit_mode": "0",
        "privilegeLevel": "1",
        "macAddr": "00:04:96:97:D1:84",
        "primary_slot": "1",
        "max_ports_per_slot": "256",
        "reboot_loop_thresh": "0",
        "hc_level": null,
        "printf_level": null,
        "status": "SUCCESS",
        "sysName": "X460G2-48t-10G4",
        "sysObjectID": "1.3.6.1.4.1.1916.2.200",
        "default_sysname": "1",
        "swVersion": " ExtremeXOS version 21.1.0.10 21.1 by dhammers on Tue
Sep 29 10:18:36 EDT 2015",
        "benchmark_mode": null,
        "trace_level": null,
        "bootTime": "Wed Sep 30 23:21:59 2015\n",
        "bootCount": "613",
        "masterCard": "Switch",
        "currTime": "Thu Oct  1 20:02:46 2015\n",
        "sysContact": "support@extremenetworks.com, +1 888 257 3000",
        "_module": "dm",
        "nextBoot": "None scheduled",
        "sysLocation": "",
        "backupCard": "Switch",
        "sysServices": "79",
        "sysType": "X460G2-48t-10G4",
        "sysDescr": "ExtremeXOS (X460G2-48t-10G4) version 21.1.0.10 21.1 by
dhammers on Tue Sep 29 10:18:36 EDT 2015"
      }
    ],
    "class": "dm_system",
    "module": "dm"
  }
}
```

Retrieving System Information from the vlan.show_ports_info Object

Request

```
{"method":"cli","id":"10","jsonrpc":"2.0","params":["debug cfgmgr show next
vlan.show_ports_info form port=None portList=1-2"]}
```

Response

```
{
  "jsonrpc": "2.0",
  "id": "10",
  "result": {
    "data": [
      {
        "noLinkUpChanges": "0",
        "tag": null,
        "displayString": null,
        "message": "",
        "port": "1",
        "qosProfile": "0",
        "noProtocols": "1",
        "portList": "1-2",
        "noVlansOnPort": "1",
        "noStpsOnThePort": null,
        "jumboSize": "9216",
        "diagTestStatus": "1",
        "driveStrength": "0",
        "status": "MORE",
        "elsmLinkState": "2",
        "iqosProfile": null,
        "ldShareMaster": null,
        "_module": "vlan",
        "lsAlgorithm": null,
        "ethOamLinkState": "2",
        "linkState": "0",
        "lsEnabled": "0",
        "flags": "Em------e--fMB---x-"
      },
      {
        "noLinkUpChanges": "0",
        "tag": null,
        "displayString": null,
        "message": "",
        "port": "2",
        "qosProfile": "0",
        "noProtocols": "0",
        "portList": "1-2",
        "noVlansOnPort": "0",
        "noStpsOnThePort": null,
        "jumboSize": "9216",
        "diagTestStatus": "1",
        "driveStrength": "0",
        "status": "SUCCESS",
        "elsmLinkState": "2",
        "iqosProfile": null,
        "ldShareMaster": null,
        "_module": "vlan",
        "lsAlgorithm": null,
        "ethOamLinkState": "2",
        "linkState": "0",
        "lsEnabled": "0",
        "flags": "Em------e--fMB---x-"
      }
    ],
    "class": "show_ports_info",
    "module": "vlan"
  }
}
```

Python Method

Python scripting is useful for creating ad hoc behaviors not covered by CLI commands. Management systems, such as Ansible, use 'on device' Python scripting as a method for network management automation. The JSONRPC Python method provides a mechanism for a script to be sent to a device and processed 'inline'. There is no need to transfer a script file to the switch before running the script. Popular tools, such as cURL or Postman, can be used to communicate with a switch. Python script lines may be terminated by newline '\n' or semicolon ';' as line separators.

For the Python method, only two fields are returned:

- stdout—contains any script output sent to stdout.
- stderr—contains any script output send to stderr.

Example

```
{"method":"python","id":"10","jsonrpc":"2.0","params":["print\nprint 'Hello
World'\nprint 'How are ya?'"]}
```

Response

```
{
  "jsonrpc": "2.0",
  "id": "10",
  "result": {
    "stderr": "",
    "stdout": "\nHello World\nHow are ya?\n"
  }
}
```

Using cURL

Some observations when using cURL:

- Single quotes on print do not work.
- Escaped double quotes are needed.

Failed example:

Request

```
curl --user admin:pass -k -i -X POST -H "Content-Type:
application/json; indent=4" \
-d '{"jsonrpc": "2.0","method": "python","params":
["print;print('hello world') "],"id": "1"}'
http://10.68.67.56/jsonrpc/
```

Response

```
curl: (3) [globbing] unmatched close brace/bracket in column 9
HTTP/1.1 200 OK
```

```
Content-Type: application/json
Content-Length: 212
Date: Thu, 01 Oct 2015 10:35:29 GMT
Server: ::

{
  "error": {
    "code": -32700,
    "data": null,
   "message": "ParseError: Unterminated string starting at: line 1 column 48
(char 48)",
    "name": "ParseError"
  },
  "id": null,
  "jsonrpc": "2.0"
```

Successful example with escaped double quotes:

Request

```
curl --user admin:pass -k -i -X POST -H "Content-Type:
application/json; indent=4" \
 -d '{"jsonrpc": "2.0","method": "python","params":
["print;print(\"hello world\") "],"id": "1"}'
http://10.68.67.56/jsonrpc/
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 107
Date: Thu, 01 Oct 2015 10:35:48 GMT
Server: ::

{
  "id": "1",
  "jsonrpc": "2.0",
  "result": {
    "stderr": "",
    "stdout": "\nhello world\n"
  }
}
```

Using Postman

With Postman, single quotes work.

3 Bug Fixes and Other Updates

None.

4 Known Limitations

None.