CALIENT

3D MEMS

Move the light, not the fiber

Software Application Interfaces Guide

Confidential and Proprietary Information

CALIENT Technologies
25 Castilian Drive
Goleta, CA 93117 USA
Phone: +1.805.562.5500
www.calient.net

Service and Support

CALIENT offers a wide range of product support programs including installation support, repair services, maintenance services and technical training.

If you need technical assistance with CALIENT’s products, please visit our automated customer support portal at http://support.calient.net or email support@calient.net.

If you are experiencing a service-affecting emergency, please contact us on the following numbers:

Within US: 1.877.682.1160
International: International Call Prefix + Country Code + 1.877.682.1160

If your call is not answered immediately, please leave a message. Messages are retrieved continuously.

Document Part Number: 460179-00, Rev. B

Revision History

Date	Version	Description	Author(s)
09/15/2014	A1	Initial draft of overview describing SW interfaces supported by S320	V. Thattai
03/02/2015	A2	Add REST API information	I. Chaube
03/09/2015	A3	Add info about reusing REST cookies	V. Thattai
03/11/2015	A4	Revise section 3.3.1 login information	S. Jayaswal
04/24/2015	A5	Change Doc. Part No. from 460171-00 to 460179-00 (old no. was already assigned to another doc)	T. Schilz
07/27/2015	A6	Update WebGUI screenshots	T. Schilz
10/08/2015	A7	Add new REST Bulk commands	T. Schilz
01/13/2017	B	Update sections 3.1.2, 3.2.2, 3.5 and 3.6; add section 4	T. Schilz

Table of Contents

List of Figures

List of Tables

PREFACE

The *Software Application Interfaces Guide* provides information on the interfaces supported by the CALIENT S320 and S160 Optical Circuit Switch (OCS).

AUDIENCE

The *Software Application Interfaces Guide* is written for both the network operations center personnel and field service personnel who configure, provision and monitor the equipment. It is assumed that this audience is familiar with TL1, REST, SNMP and OpenFlow protocols.

1 INTRODUCTION

The CALIENT OCS can be managed, controlled and monitored by a variety of application interfaces. Each of the interfaces listed below is supported by the OCS and described in this document:

- TL1
- WebGUI
- REST
- SNMP (v1, v2c, v3)
- OpenFlow (1.0, 1.3)
- SSH/SCP (Secure Shell/Secure Copy)
- RPC (Remote Procedure Call)
- CORBA

Dual Gigabit Ethernet Ports provide the IP interface over which these protocols run. The OCS also has a Serial Console Port that is used for initial setup and configuration of the system.

2 SOFTWARE ARCHITECTURE

High-level software architecture, including application and system-level building blocks, is depicted in Figure 1.

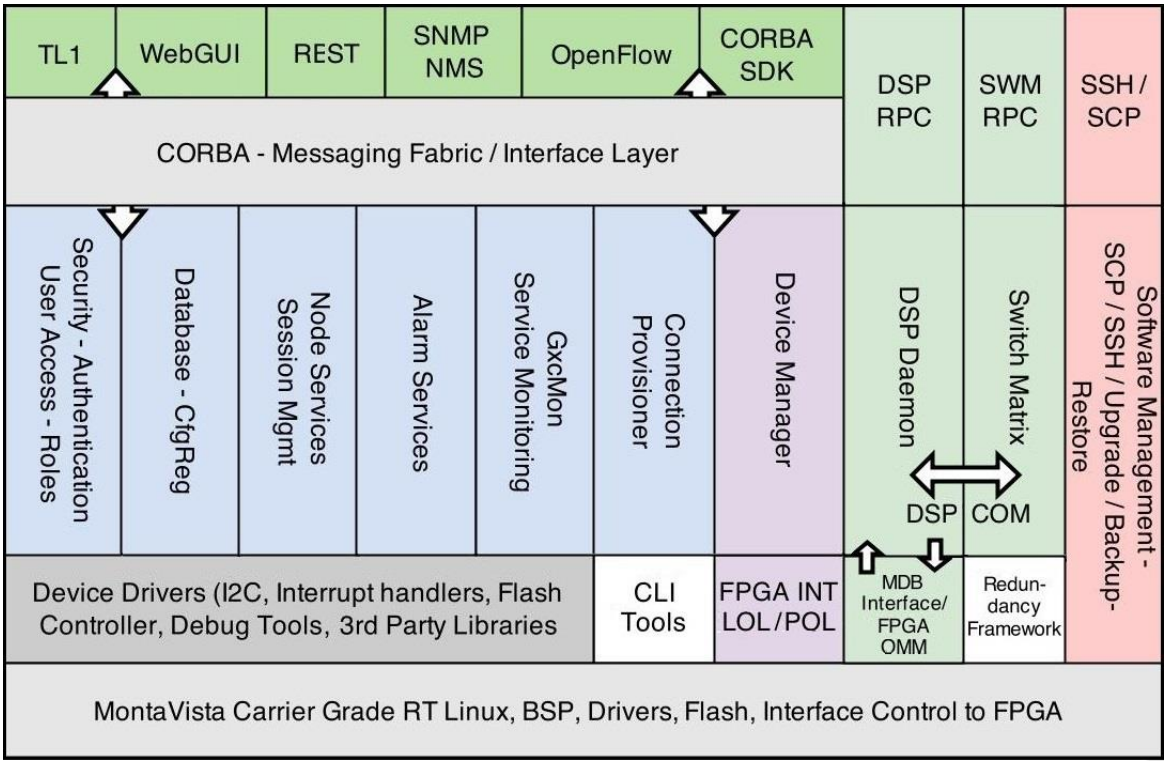


Figure 1 – Software Architecture Diagram

3 SOFTWARE INTERFACES

At the high level, the following software interfaces are available for programming and/or provisioning the CALIENT OCS:

- TL1
- WebGUI
- REST
- SNMP/NMS
- OpenFlow
- SSH/SCP
- DSP RPC
- Switch Matrix RPC
- CORBA SDK

3.1 TL1

Transaction Language 1 (TL1) is the only standard, cross-vendor command line interface (CLI). It is well documented and has a well-defined syntax, making it suitable for both humans and computing devices.

The OCS supports a TL1 interface based on Telcordia standards including GR-831, GR-199-CORE and GR-833-CORE.

A TL1 user ID can run multiple sessions concurrently, up to the maximum of 20. The TL1 interface is accessible through:

- Telnet
- SSH

For detailed information on TL1 management, please contact marketing@calient.net to obtain the current version of the *CALIENT Optical Circuit Switch (OCS) TL1 Reference Guide*.

3.1.1 Connecting with TL1 via Telnet

To connect via TELNET, specify the node IP address and the port socket that provides the communication path. Two ports are used: 3082 and 3083. Each port represents a mode of operation. For example:

- telnet 192.168.110.13 3082
- telnet 192.168.110.13 3083

Table 1 below describes the differences between the two modes of operation.

Table 1 – TL1-RAW Mode vs. TL1-TELNET Mode

Port Socket	Mode	Description
3082	TL1-RAW	• With this mode, the user does not interact with the TL1 Agent, and does not receive the *agent>* prompts at the console. • When provisioning in bulk, select this mode for executing automated scripts. • Online help is not available with this mode.
3083	TL1-TELNET	• Typically, this mode is used for individual configuration or provisioning. This mode enables the user to interact with the TL1 Agent through the console, and receive prompts and messages. • Online help is available with this mode.

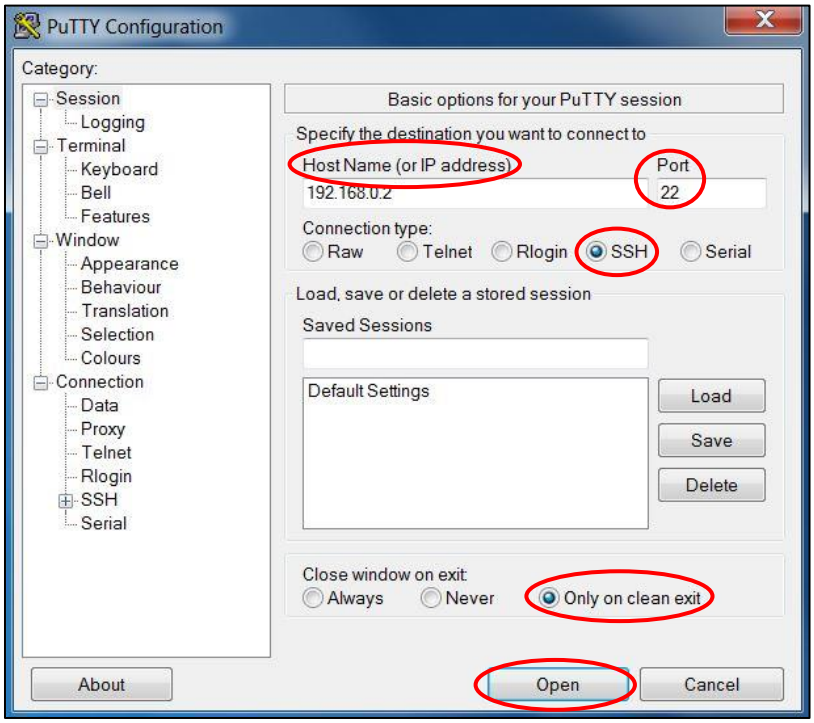
3.1.2 Connecting with TL1 via SSH

To connect to the system via SSH, CALIENT recommends using an SSH client, such as PuTTY. PuTTY is a free, open-source SSH and Telnet software application for Windows (and some Unix-like platforms) that can be used to make an SSH connection to your device. Free PuTTY software can be downloaded at http://www.putty.org.

3.1.2.1 Connecting Using PuTTY

The following procedure describes how to connect to the OCS using PuTTY:

1. Click the PuTTY icon on your desktop to open the application. The **PuTTY Configuration** screen will appear.



2. Enter the IP address of the switch in the **Host Name (or IP address)** field of the screen.

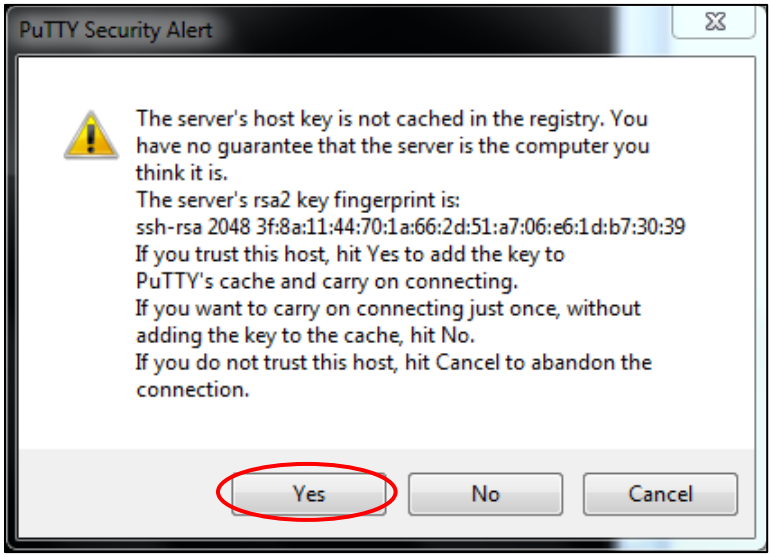
Note

The IP address shown in the image in Step 1 is for illustrative purposes only and not intended for use on your system.

3. Verify that the **Port** field shows **22** as the port number; if it doesn't, enter 22 and *leave it as the default value* for the field.
4. Verify that the **Connection type** is **SSH**. If not, click the SSH radio button to select it, and *leave it as the default value* for the setting.
5. Click the **Only on clean exit** radio button in the **Close window on exit:** field of the screen to select it.
6. Click the **Open** button to start the SSH session.

Note

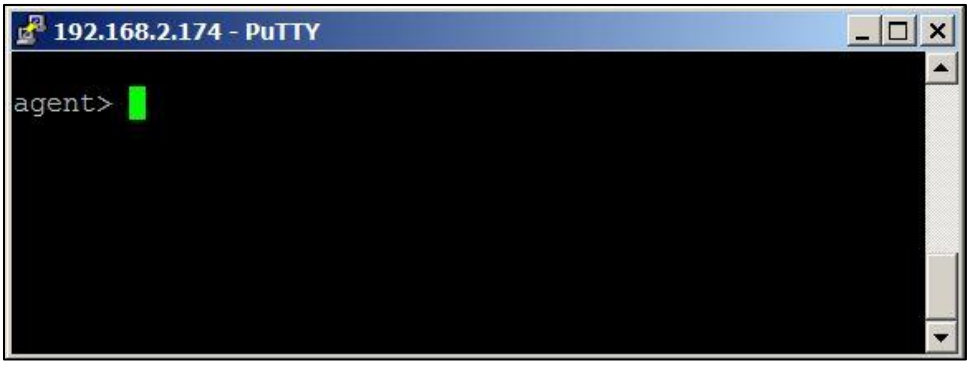
The first time you connect to the OCS from your computing device, a PuTTY Security Alert screen will appear, warning that you are trying to connect to an unknown host. Click the **Yes** button at the bottom of the screen to accept the connection.



Once the SSH connection is open, the PuTTY login screen will appear, prompting you to enter your user ID.

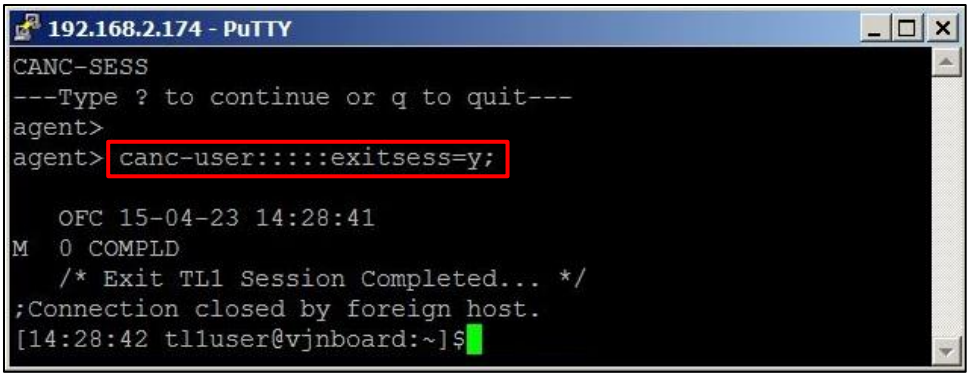
7. Type in `tl1user` and press Enter on your keyboard.

8. Press Enter on your keyboard. The `agent>` prompt will appear.



9. Type in the commands for configuring or retrieving information about the system.
10. When you have finished your TL1/SSH session, type in the following command to gracefully exit from it:

```
agent> canc-user:::::exitsess=y;
```



3.2 WebGUI

The OCS Web-based GUI (Graphical User Interface) can be used to configure system parameters, add, view and delete cross-connects, assign ports, manage users and monitor faults.

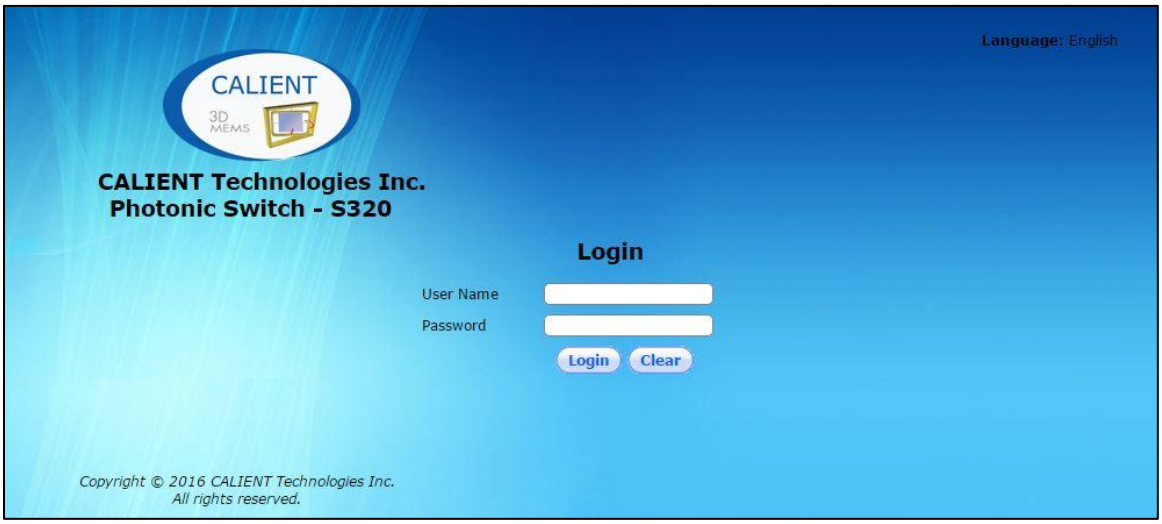
3.2.1 Logging In

The following procedure describes how to log into the WebGUI:

1. Enter the IP address of the switch in your web browser. The WebGUI will launch, and the WebGUI Login screen will appear in your browser window.

Note

The OCS ships with a default IP address of 192.168.0.2. You may need to change the default to suit your network configuration. For information on setting the IP address, please refer to the *CALIENT OCS Quick Start Guide* and/or the Node Configuration section of the *CALIENT OCS TL1 Reference Guide*.



2. Log into the GUI using the default username (**admin**) and password (**pxc*****).

For more information on how to use the OCS WebGUI, please contact marketing@calient.net to access the current version of the *OCS WebGUI Quick Start Guide* or online video tutorials for the OCS WebGUI.

3.2.2 SSL (HTTPS) Interface for Web Access

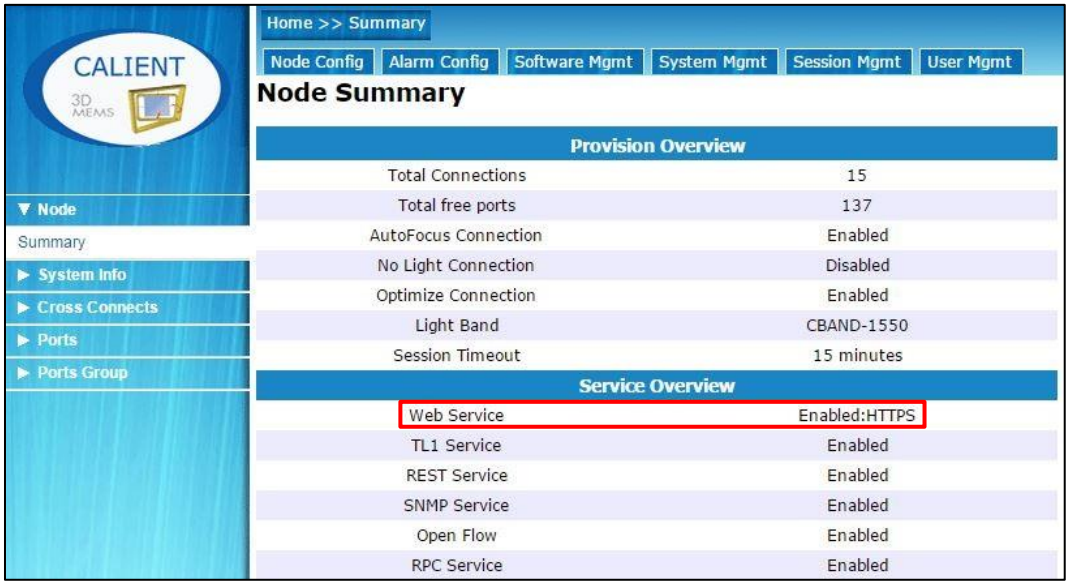
A key feature of the OCS WebGUI is an SSL (HTTPS) interface for secure Web access. This interface can be enabled or disabled over the Web from the **Node Configuration** page, located at **Node** > **Summary** > **Node Config**. Supported configuration options include:

- HTTP (only) – the default setting for Web access
- HTTPS (only)
- BOTH (http and https)

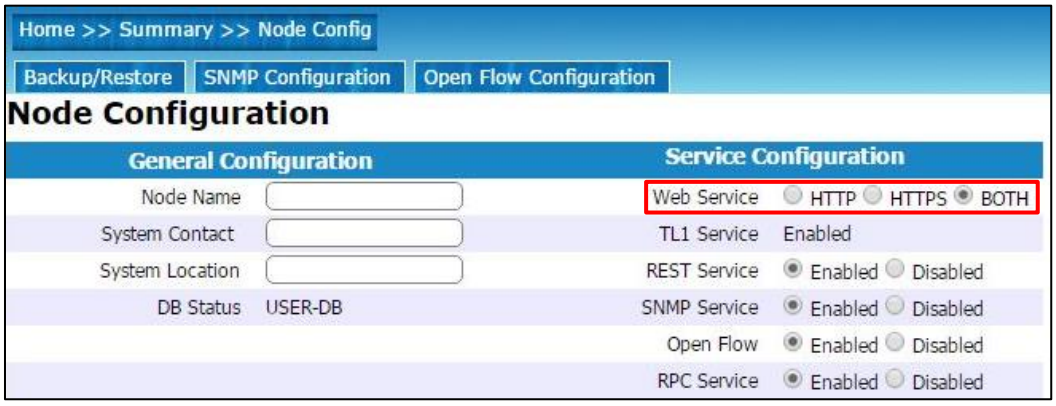
Note

Upon changing the Web configuration, Web services will be down momentarily.

The **Service Overview** section on the **Node Summary** page of the WebGUI shows the Web Service configured on the system. In the example below, HTTPS Web Service is enabled.



The currently configured Web Service can be changed by resetting the options provided in the **Service Configuration** section on the **Node Configuration** page of the WebGUI (located at **Node** > **Summary** > **Node Config**). In the example below, HTTPS has been changed to HTTPS *and* HTTP by clicking the BOTH radio button to select that option from those listed for Web Service.



3.2.2.1 Enabling HTTPS on the OCS

In order to enable HTTPS on the OCS, you will need to obtain an SSL certificate from a valid organization like Verisign or Thawte. You can also generate a self-signed SSL certificate for testing purposes.

Key files can be generated on the OCS or on a different machine. If the files are generated on a different machine, they will have to be copied from the machine to the **/etc/ssl/** directory on the OCS.

The following naming convention MUST be adhered to when naming key files for the S320 OCS:

- The private key file MUST be named as **private.key**
- The certificate signing request MUST be named as **CSR.csr**
- The webserver certificate file MUST be named as **CRT.crt**
- The PEM file MUST be named as **PEM.pem**
- All of these files MUST be stored in the **/etc/ssl/** directory on the OCS
- Only `root` should have read permissions for these files:

```
chown root:root /etc/ssl/PEM.pem

chmod 0600 /etc/ssl/PEM.pem
```

Additional information about .pem, .csr and .crt extensions can be found here.

If keys and certificates already exist, follow the above naming convention and permissions for the certificate and key files.

Copy files to the **/etc/ssl/** directory and restart Web services:

```
WebServices.sh stop

WebServices.sh start
```

3.2.2.2 Generating the Private Key and CSR

There are several encryption algorithms that can be used based on the level of encryption and security required.

The following example illustrates how to generate a private key and CSR using the RSA:2048-bit encryption algorithm:

```
/opt/calient/GXCP/lighttpd/sbin/openssl req -nodes -newkey rsa:2048
-keyout /etc/ssl/private.key -out /etc/ssl/CSR.csr

Generating a 2048 bit RSA private key

.+++

.+++

writing new private key to '/etc/ssl/private.key'

-----

You will be asked to enter information that will be incorporated
into your certificate request.

The information you enter is called a Distinguished Name (DN).

Although there are quite a few fields, some can be left blank.

Certain fields will have a default value.

If you enter a period (.), the field will be left blank.
```

Country Name (2-letter code) [AU]: US

State or Province Name (full name) [Some-State]: California

Locality Name (e.g., city) []: Goleta

Organization Name (e.g., company) [Internet Widgits Pty Ltd]: Calient Networks

Organizational Unit Name (e.g., section) []: Engineering

Common Name (e.g., server FQDN or YOUR name) []: Nevada.row9.SW1854

Email Address []: admin@datacenter.com

```
Also, enter the following "extra" attributes to be sent with your
certificate request:
```

A challenge password []: Calient123Secret

An optional company name []: Calient Technologies

3.2.2.3 Generating a Self-Signed SSL Certificate

For testing purposes, you can generate a self-signed SSL certificate that is valid for 1 year using the `openssl` command, as shown below.

Instead of signing it yourself, you can also generate a valid SSL certificate from THAWTE or any other signing authority.

```
/opt/calient/GXCP/lighttpd/sbin/openssl x509 -req -days 365 -in
/etc/ssl/CSR.csr -signkey /etc/ssl/private.key -out /etc/ssl/CRT.crt
```

3.2.2.4 Creating a PEM file

The following example shows how to create a PEM file:

```
cat /etc/ssl/private.key /etc/ssl/CRT.crt > /etc/ssl/PEM.pem
```

3.2.2.5 Checking the /etc/ssl Directory and File Permissions

The following example shows how to check the **/etc/ssl** directory and file permissions:

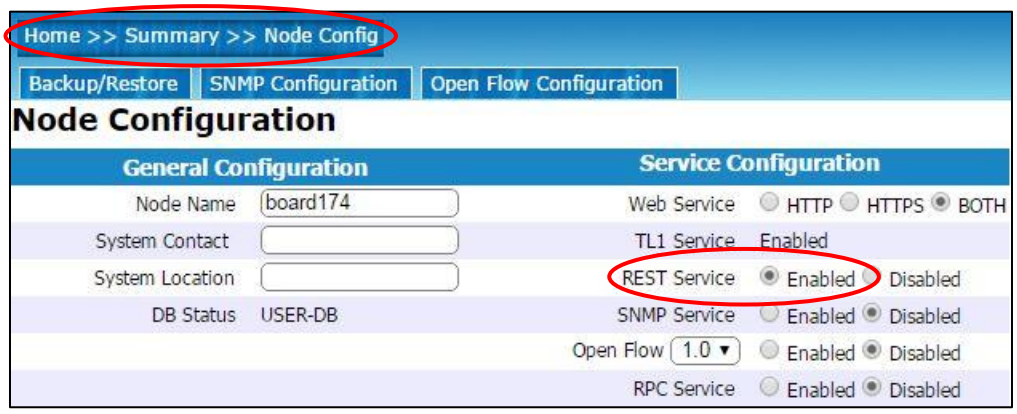
```
chown root:root /etc/ssl/PEM.pem

chmod 0600 /etc/ssl/PEM.pem
```

3.3 REST

CALIENT has extended the REST (Representational State Transfer) API framework to provide additional interfaces for configuring the OCS, and for accessing alarm, event and port information for the switch.

The following procedure describes how to enable REST service on the OCS:

1. Log into the OCS.
2. Navigate the following path: **Node** > **Summary** > **Node Config**. The **Node Configuration** screen will open.



3. Click the **Enabled** radio button next to **REST Service** in the **Service Configuration** section of the **Node Configuration** screen to enable REST.

3.3.1 Logging In

Accessing the REST interfaces on the OCS can be done through either the command line interface (CLI) or the WebGUI. In either case, authentication (i.e., entering a valid username and password) is required by the system before a user can issue commands.

3.3.1.1 Accessing REST via the CLI

Following is an example of an attempt by an authorized user to post cross-connect data using the CLI:

Command:

```
curl -i -X POST --data id=add --data in=1 --data out=2 --data dir=Bi -
-user admin:pxc*** 'http://192.168.0.2/rest/crossconnects/?id=add'
```

Response:

```
HTTP/1.1 200 OK
X-Powered-By: PHP/5.3.6
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, DELETE
Set-Cookie: PHPSESSID=i7o7kjmoehavifhn230u9tlr64; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
check=0
Pragma: no-cache
Content-type: text/html
Transfer-Encoding: chunked
```

```
Date: Thu, 12 Mar 2015 00:08:39 GMT
Server: lighttpd/1.4.29

[{"status":"1","msg":"OK","description":"Cross Connection added
successfully!"}]
```

3.3.1.2 Accessing REST via the WebGUI

The REST interface API is accessible from the WebGUI home page, as described below:

1. Click the REST API button at the top of the screen.
2. Clicking any listed section will display the available APIs for that section.

3.3.2 REST Commands

The REST commands described in the following subsections can be used to manage the OCS. Three method types are used for issuing REST commands: GET, POST and DELETE.

> **Note**
>
> It is recommended that the cookies generated when CURL sessions are created be reused to extend the number of simultaneous sessions on the system. Please refer to section 3.3.3 for specific details.

3.3.2.1 Add Cross Connection

Method: POST

URL: http://OCS IP address/rest/crossconnects/

Required Parameters:

```
id=add, in=<in port>, out=<out port>
```

Optional Parameters:

```
group=<group name>, conn=<conn name>, dir=<direction>, band=<band>
```

Optional Parameter Values:

```
groupName and ConnectionName are strings with max = 35 characters

dir supported values = {Uni, Bi}

band supported values = {'C' for CBAND, 'O' for OBAND, 'L' for LBAND,
'S' for SBAND, 'W' for OBAND }
```

Examples:

If the attempt to add a cross-connection is successful, the status information returned by the switch will be “1”.

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X POST --data
id=add --data in=1.1.1 --data out=1.1.2 --user admin:pxc***
'192.168.0.2/rest/crossconnects/'
```

Response:

```
[{"status":"1","msg":"OK","description":"Cross Connection added
successfully!"}]
```

If the attempt to add a cross-connection fails, the status information returned by the switch will be “0”:

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X POST --data
id=add --data in=1.1.1 --data out=1.1.2 --user admin:pxc***
'192.168.0.2/rest/crossconnects/'
```

Response:

```
[{"status":"0","msg":"Exist","description":"Connection already
exists!"}]
```

Example of a bi-directional connection:

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10   -X POST --data
id=add --data in=92 --data out=93 --data dir=Bi --user admin:pxc***
'192.168.0.2/rest/crossconnects/'

curl -X POST --data id=add --data in=3.3.5 --data out=3.3.1
--data dir=Bi --user <username>:<password>  '<URI IP
address>/rest/crossconnects/'
```

Response:

```
[{"status":"1","msg":"OK","description":"Cross Connection added
successfully!"}]
```

3.3.2.2 Bulk Add

Method: POST

URL: http://OCS IP address/rest/crossconnects/?id=badd

Required Parameters: id=badd

Optional Parameters: None

Post Data: JSON array format

Sample Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -H "Content-
Type: application/json" -X POST -d'[{
"id":"add","in":"1","out":"2"},{ "id":"add", in":"3","out":"6"}]' -
-user admin:pxc***
'http://192.168.102.198/rest/crossconnects/?id=badd'
```

Sample Response:

```
[{"id":"madd","in":"1","out":"2","response":{"status":"1","msg":"OK
","description":"Cross Connection added
successfully!"}},{"id":"madd","in":"3","out":"6","response":{"statu
s":"1","msg":"OK","description":"Cross Connection added
successfully!"}}]
```

3.3.2.3 Delete Cross Connection

Method: DELETE

URL: http://OCS IP address/rest/crossconnects/

Parameter:

```
?conn=<connection ID>
```

Example:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X DELETE --user
admin:pxc*** 'http://192.168.0.2/rest/crossconnects/?conn=1.7.5>1.7.5'
```

Response:

```
[{"status":"1","msg":"Deleted","description":"Cross Connection deleted
successfully!"}]
```

3.3.2.4 Bulk Delete

Method: POST

URL: http://OCS IP address/rest/crossconnects/?id=bdel

Required Parameters: id=bdel

Optional Parameters: None

Post Data: JSON array format

Sample Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -H "Content-
Type: application/json" -X DELETE -d'[{ "conn":"1>2”},{
"conn":"3>6" }]' --user admin:pxc***
'http://192.168.102.198/rest/crossconnects/?id=bdel'
```

Sample Response:

```
[{"conn":"1-
2","response":{"status":"1","msg":"OK","description":"Cross
Connection(s) deleted
successfully!"}},{"conn":"3>6","response":{"status":"1","msg":"OK",
"description":"Cross Connection(s) deleted successfully!"}}]
```

3.3.2.5 Cross Connection List

Method: GET

URL: http://<OCS IP address>/rest/crossconnects/

Parameter:

```
?id=list
```

Example (CLI):

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X GET --user
admin:pxc*** 'http://192.168.0.2/rest/crossconnects/?id=list'
```

Response:

```
[{"name":"1.1.1>1.1.2","group":"SYSTEM","connid":"1.1.1>1.1.2","dir":"
UNI","band":"CBAND","half1":{"conn":"1.1.1>1.1.2","inp":"4.25","outp":
"2.25","loss":"2.00","alarm":"CL","as":"UMA","os":"RDY","oc":"OK"}},{"
name":"003-004","group":"SYSTEM","connid":"1.1.3-
1.1.4","dir":"BI","band":"CBAND","half1":{"conn":"1.1.3>1.1.4","inp":"
4.25","outp":"2.25","loss":"2.00","alarm":"CL","as":"UMA","os":"RDY","
oc":"OK"},"half2":{"conn":"1.1.4>1.1.3","in":"4.25","outp":"2.25","los
s":"2.00","alarm":"CL","as":"UMA","os":"RDY","oc":"OK"}}]
```

Example (WebGUI):

http://<OCS IP address>/rest/crossconnects/?id=list

3.3.2.6 Cross Connection Detail

Method: GET

URL: http://<OCS IP address>/rest/crossconnects/

Parameter:

```
?id=detail&conn=<conn ID>
```

Example:

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X GET --user
admin:pxc***
'http://192.168.0.2/rest/crossconnects/?id=detail&conn=2.8.5-2.8.1'
```

Response:

```
[{"name":"2.8.5-2.8.1","group":"SYSTEM","connid":"2.8.5-
2.8.1","dir":"BI","band":"C","autofocus":"Enabled","nolight":"Disabled
","half1":{"conn":"2.8.5>2.8.1","inp":"4.25","outp":"4.25","loss":"4.2
5","alarm":"CL","as":"UMA","os":"RDY","oc":"OK"},"half2":{"conn":"2.8.
1>2.8.5","in":"4.25 ","outp":"2.25","loss":"2.00
","alarm":"CL","as":"UMA","os":"RDY","oc":"OK"}}]
```

3.3.2.7 Port Summary

Method: GET

URL: http://<OCS IP address>/rest/ports/

Parameter:

```
?id=summary
```

Example:

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X GET --user
admin:pxc*** 'http://192.168.0.2/rest/ports/?id=summary'
```

Response:

```
[{"port":"1.1.1","inalias":"RTYRTYERTY","outalias":"RETYTT","power":"1
2.50","connid":"1.1.1>1.1.2","conn":"1.1.1>1.1.2"},{"port":"1.1.2","in
alias":"","outalias":"","power":"12.50","connid":"","conn":""},{"port"
:"1.1.3","inalias":"","outalias":"","power":"12.50","connid":"1.1.3-
1.1.4","conn":"003-
004"},{"port":"1.1.4","inalias":"","outalias":"","power":"12.50","conn
id":"1.1.3-1.1.4","conn":"003-
004"},{"port":"1.1.5","inalias":"","outalias":"","power":"12.50","conn
```

id":"","conn":""},{"port":"1.1.6","inalias":"","outalias":"","power":"
12.50","connid":"","conn":""},…]

3.3.2.8 Port Detail

Method: GET

URL: http://<OCS IP address>/rest/ports/

Parameter:

```
?id=detail&port=<port number>
```

Example:

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X GET --user
admin:pxc*** 'http://192.168.0.2/rest/ports/?id=detail&port=1.1.1'
```

Response:

[{"inpower":"-0.0","incircuit":"1.1.1>1.1.2","inas":"IS","inos":"OOS",
"inoc":"NOHW","inowner":"TRANSIT","inalias":"RTYRTYERTY","inopth":
"13.0","inopt":"-15.0","inoptc":"-17.0","outpower":"-90.0","outcircuit":
"", "outas":"OOS-NP","outos":"OOS","outoc":"NOHW","outowner": "NONE",
"outalias":"RETYTT","outopwdh":"-18.0","outopwct":"-20.0"}]

3.3.2.9 Alarms

Method: GET

URL: http://<OCS IP address>/rest/alarms/

Required Parameter:

```
?id=active
```

Optional Parameter:

```
?id=active &type=<CRITICAL or MAJOR or MINOR>
```

Examples:

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X GET --user
admin:pxc*** 'http://192.168.0.2/rest/alarms/?id=active'
```

Response:

[{"id=":"9","severity":"CR","time":"02\/23\/15
17:02:36","action":"SET","condition":"ENV","entity":"ENVMON","descript
ion":"Output OMM Interrupt

Failure"},{"id=":"8","severity":"CR","time":"02\/23\/15
17:02:34","action":"SET","condition":"ENV","entity":"ENVMON","descript
ion":"Input OMM Interrupt
Failure"},{"id=":"7","severity":"MJ","time":"02\/23\/15
17:02:32","action":"SET","condition":"ENV","entity":"ENVMON","descript
ion":"Power Feed Unit B
Failed"},{"id=":"6","severity":"MJ","time":"02\/23\/15
17:02:30","action":"SET","condition":"ENV","entity":"ENVMON","descript
ion":"Power Feed Unit A
Failed"},{"id=":"1","severity":"MJ","time":"02\/23\/15
17:01:38","action":"SET","condition":"ENV","entity":"I2C","description
":"Fan Control Unit Access Failed"}]

Example (MAJOR alarm):

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X GET --user
admin:pxc*** 'http://192.168.0.2/rest/alarms/?id=active&type=MAJOR'
```

Response:

[{"id=":"7","severity":"MJ","time":"02\/23\/15
17:02:32","action":"SET","condition":"ENV","entity":"ENVMON","descript
ion":"Power Feed Unit B
Failed"},{"id=":"6","severity":"MJ","time":"02\/23\/15
17:02:30","action":"SET","condition":"ENV","entity":"ENVMON","descript
ion":"Power Feed Unit A
Failed"},{"id=":"1","severity":"MJ","time":"02\/23\/15
17:01:38","action":"SET","condition":"ENV","entity":"I2C","description
":"Fan Control Unit Access Failed"}]

3.3.2.10 Alarms/Event

Method: GET

URL: http://<OCS IP address>/rest/alarms/

Required Parameter:

```
?id=all
```

Optional Parameters:

```
?id=all &type=<CRITICAL or MAJOR or MINOR> &class=<COM,CRS,ENV,EQPT,
SECU>
```

Example (CLI):

Command:

```
curl -i --cookie PHPSESSID=b80tsidd01les0f9ji9k4dho10 -X GET --user
admin:pxc***
'http://192.168.0.2/rest/alarms/?id=all&type=MAJOR&class=ENV'
```

Response:

```
[{"id=":"7","severity":"MJ","time":"02\/23\/15
17:02:32","action":"SET","condition":"ENV","entity":"ENVMON","descript
ion":"Power Feed Unit B
Failed"},{"id=":"6","severity":"MJ","time":"02\/23\/15
17:02:30","action":"SET","condition":"ENV","entity":"ENVMON","descript
ion":"Power Feed Unit A
Failed"},{"id=":"1","severity":"MJ","time":"02\/23\/15
17:01:38","action":"SET","condition":"ENV","entity":"I2C","description
":"Fan Control Unit Access Failed"}]
```

Example (WebGUI):

http://<OCS IP address>/rest/alarms/?id=all&type=MAJOR&class=ENV

3.3.3 Reusing REST Cookies

It is recommended that the cookies generated when CURL sessions are created be reused to extend the number of simultaneous sessions on the system. If cookies are not reused, a new session is created on the OCS for every request it receives; when 9 such sessions are reached (the maximum sessions allowed), subsequent requests are denied. Reusing CURL session cookies will prevent reaching the maximum session limit.

As shown in the following example, the first invocation returns the **PHPSESSID attribute**, which **can be reused** in subsequent calls.

Command:

```
curl -i  -X GET --user admin:pxc***
'http://192.168.0.2/rest/alarms/?id=active'
```

Response:

```
HTTP/1.1 200 OK
X-Powered-By: PHP/5.3.6
Access-Control-Allow-Origin: *
Set-Cookie: PHPSESSID=0kkn1llpomg64u9iuf3tmpq5u6; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
```

```
Content-Type: application/json
Transfer-Encoding: chunked
Date: Mon, 09 Mar 2015 21:31:04 GMT
Server: lighttpd/1.4.29

 [{"id=":"7","severity":"CR","time":"03\/08\/15
21:37:01","action":"SET","condition":"ENV","entity":"ENVMON","desc
ription":"Output OMM Interrupt Failure"}]
```

Command (reusing cookie from previous response):

```
curl -i --cookie PHPSESSID=0kkn1llpomg64u9iuf3tmpq5u6  -X GET --
user admin:pxc*** 'http://192.168.0.2/rest/alarms/?id=active'
```

Response:

```
HTTP/1.1 200 OK
X-Powered-By: PHP/5.3.6
Access-Control-Allow-Origin: *
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Content-Type: application/json
Transfer-Encoding: chunked
Date: Mon, 09 Mar 2015 21:31:29 GMT
Server: lighttpd/1.4.29
```

[{"id=":"7","severity":"CR","time":"03\/08\/15
21:37:01","action":"SET","condition":"ENV","entity":"ENVMON","description":"Output OMM
Interrupt Failure"}]

3.4 SNMP

The CALIENT SNMP Agent supports SNMPv1, SNMPv2c and SNMPv3. With SNMPv3 support, the CALIENT SNMP implementation is a highly secure interface. CALIENT OCS features are defined as part of Enterprise MIBs. The OCS SNMP Agent supports the following SNMP functions:

- Get/Set
- GetBulk requests for SNMP MIB-II MIBs
- Enterprise-specific MIBs
- Alarm/Event Traps

Trap versions supported are v1 Trap, v2c Trap, v2c Inform, v3 Trap and v3 Inform. CALIENT OCS Agent MIBs can be copied from the installation package located at:

/opt/calient/GXCP/net-snmp/share/snmp/mibs

Table 2 describes the functionality and configuration of the OCS's Enterprise-specific MIBs:

Table 2 – CALIENT MIBs

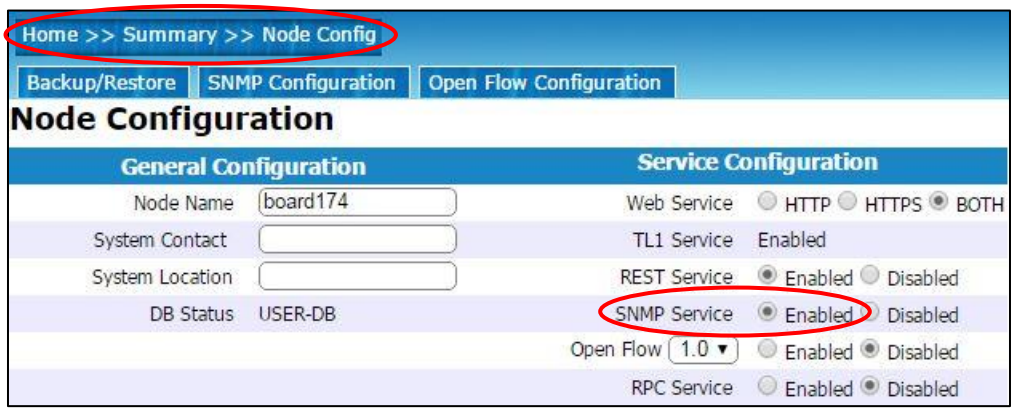
S. No.	MIB Name	Description
1	CALIENT-SMI	CALIENT SMI and top-level registrations
2	CALIENT-TC	CALIENT textual conventions
3	CALIENT-CHASSIS-MIB	MIB describing the physical elements (primarily shelves and modules) in a chassis
4	CALIENT-PORTS-MIB	MIB describing CALIENT port features
5	CALIENT-CONNECTION-MIB	MIB describing a list of transit connections provisioned in this PXC. This MIB supports CALIENT connection creation using SNMP set functionality as well.
6	CALIENT-PORT-GROUP-MIB	MIB describing a list of the port groups configured in this PXC
7	CALIENT-SW-MGMT-MIB	MIB describing a list of software versions running on various slots in the chassis
8	CALIENT-ALARM-CONF-MIB	MIB defining the various alarms supported by CALIENT devices
9	CALIENT-SERVICE-CONF-MIB	MIB describing the various services running on the CALIENT device
10	CALIENT-SCP-CONF-MIB	MIB describing the SCP configuration for installation/backup
11	CALIENT-NTP-SERVER-CONF-MIB	MIB describing the NTP configuration for the switch
12	CALIENT-USER-MGMT-MIB	MIB describing the list of users configured to access this device
13	CALIENT-SESSION-MGMT-MIB	MIB describing a list of configured user sessions
14	CALIENT-SECURITY-MIB	MIB describing support for security profiles and monitoring of access to CALIENT nodes
15	CALIENT-ENV-MIB	MIB monitoring the CALIENT environment

A list of supported CALIENT MIB OIDs can be found in the *CALIENT SNMP User Guide*. To access this document, please contact marketing@calient.net.

3.4.1 Configuring SNMP with the WebGUI

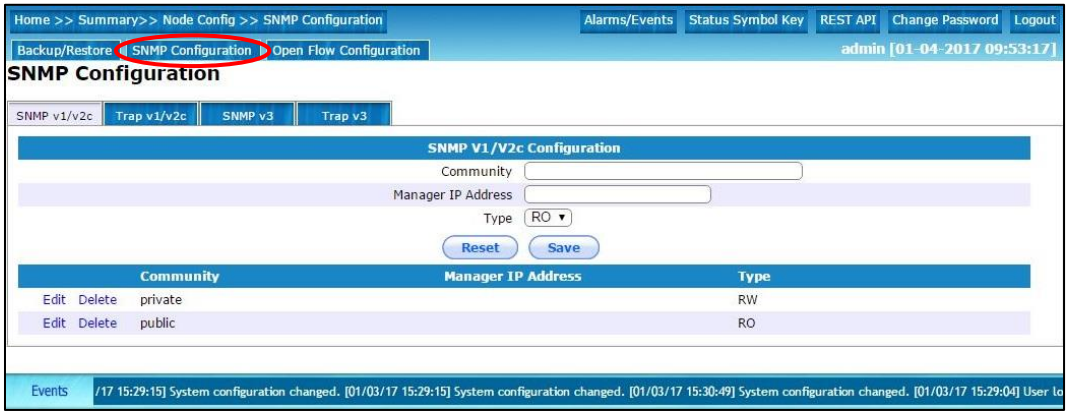
The following procedure describes how to configure SNMP on the OCS using the WebGUI:

1. Log into the OCS.
2. Navigate the following path: **Node** > **Summary** > **Node Config**. The **Node Configuration** screen will open.



3. Click the **Enabled** radio button next to **SNMP Service** in the **Service Configuration** section of the **Node Configuration** screen to enable SNMP.

   Once SNMP is enabled, additional parameters (see image below) can be configured by clicking the **SNMP Configuration** tab on the **Node Configuration** screen.



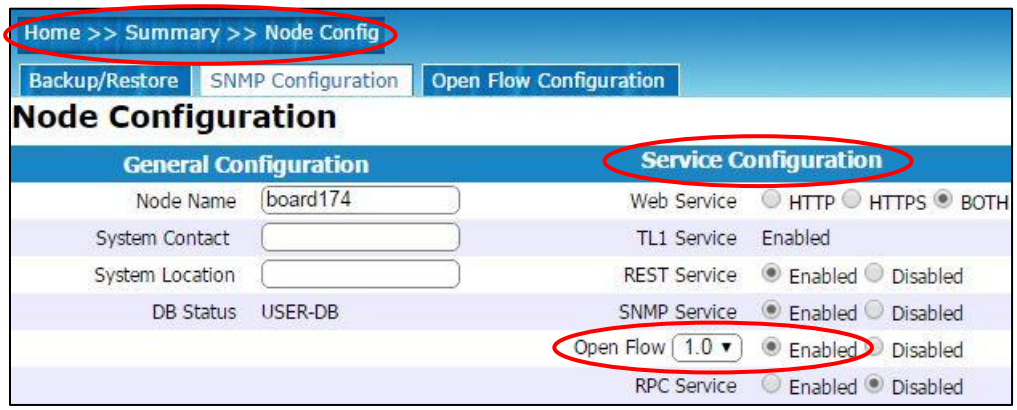
3.5 OpenFlow

The CALIENT OCS supports versions 1.0 and 1.3 of the OpenFlow communications protocol, the de facto standard for enabling software-defined networking (SDN). Sections 3.5.1 and 3.5.2 describe how to configure OpenFlow on the OCS with the WebGUI.

3.5.1 Configuring OpenFlow 1.0 with the WebGUI

The following procedure describes how to configure OpenFlow 1.0 on the OCS using the WebGUI:

1. Log into the OCS.
2. Navigate the following path: **Node** > **Summary** > **Node Config**. The **Node Configuration** screen will open.

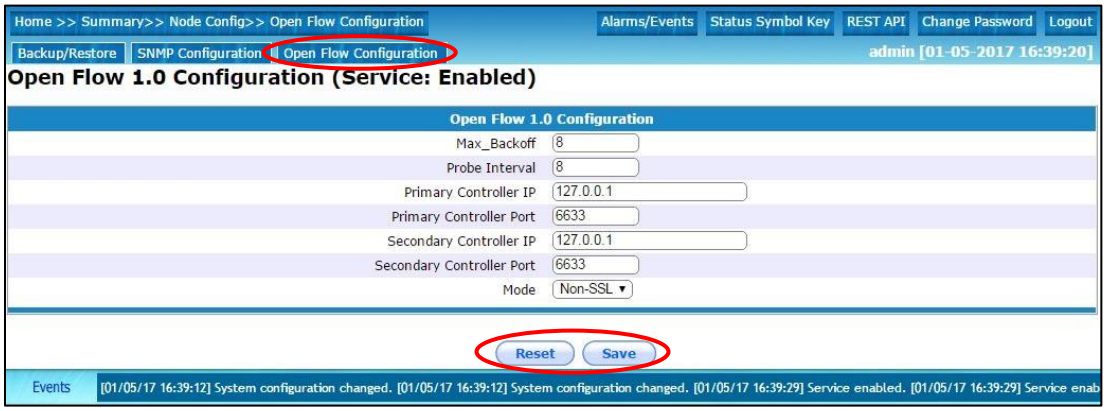


3. Locate the **Service Configuration** section of the screen.
4. Click the down arrow (▼) at the right of the Open Flow dropdown menu to display the OpenFlow versions currently supported on the OCS.
5. Click one of the listed versions (e.g., 1.0) to select it.
6. Click the **Enabled** radio button next to the Open Flow dropdown menu to enable the selected version (1.0).

3.5.1.1 Configuring Additional OpenFlow 1.0 Parameters

Once OpenFlow is enabled, additional parameters can be configured as follows:

1. Click the **Open Flow Configuration** tab on the **Node Configuration** screen. The **Open Flow 1.0 Configuration** screen will appear.

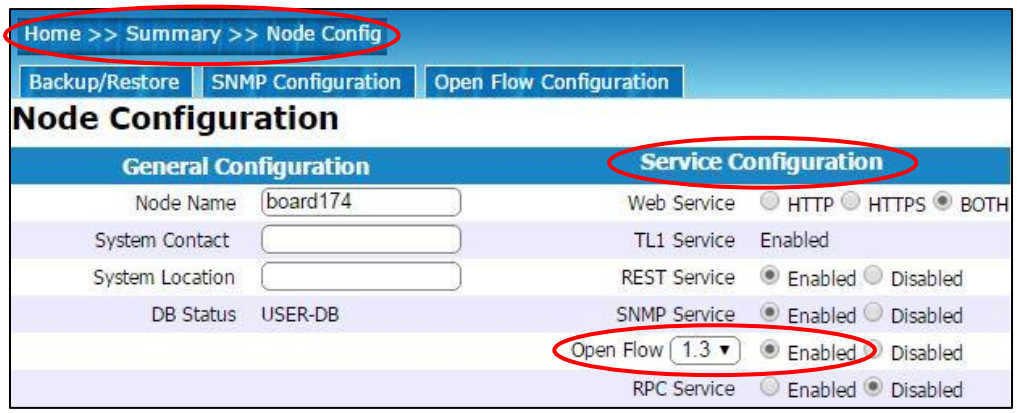


2. Enter new data, or edit existing data, in any of the fields displayed in the **Open Flow 1.0 Configuration** section of the screen.
3. Click the **Reset** button to undo any new changes, OR
4. Click the **Save** button to keep the new changes.

3.5.2 Configuring OpenFlow 1.3 with the WebGUI

The following procedure describes how to configure OpenFlow 1.3 on the OCS using the WebGUI:

1. Log into the OCS.
2. Navigate the following path: **Node** > **Summary** > **Node Config**. The **Node Configuration** screen will open.



3. Locate the **Service Configuration** section of the screen.

4. Click the down arrow (▼) at the right of the Open Flow dropdown menu to display the OpenFlow versions currently supported on the OCS.

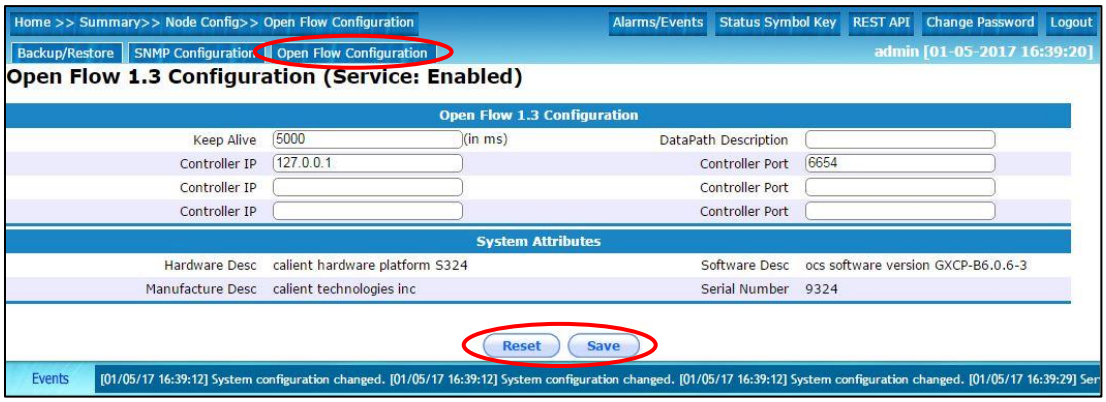
5. Click one of the listed versions (e.g., 1.3) to select it.

6. Click the **Enabled** radio button next to the Open Flow dropdown menu to enable the selected version (1.3).

3.5.2.1 Configuring Additional OpenFlow 1.3 Parameters

Once OpenFlow is enabled, additional parameters can be configured as follows:

1. Click the **Open Flow Configuration** tab on the **Node Configuration** screen. The **Open Flow 1.3 Configuration** screen will appear.



2. Enter new data, or edit existing data, in any of the fields displayed in the **Open Flow 1.3 Configuration** section of the screen.

3. Click the **Reset** button to undo any new changes, OR

4. Click the **Save** button to keep the new changes.

3.6 SSH/SCP Interfaces

SSH (Secure Shell) and SCP (Secure Copy) interfaces are provided on the OCS. Python (or another) programming language can be used to create scripts over SSH and program/configure the switch.

3.6.1 Public-Key Authentication

Public-key authentication is an alternative method of identifying yourself to a login server, instead of typing a password. Although it involves more steps than a password entry, public-key

authentication provides greater security and flexibility than a password alone provides.

Sections 3.6.1.1 and 3.6.1.2 describe how to set up the public-key authentication included with the OCS, as well as public-key authentication generated "natively" by the user.

3.6.1.1 Setting Up Public-Key Authentication

The following procedure describes how to set up public-key authentication using SSH on a Linux system:

1. Log into the computing device you'll use to access the OCS.
2. Use command-line SSH to generate a key pair with RSA (or any other algorithm).
3. Generate RSA keys from the command line by entering the following command:

   ```
   ssh-keygen -t rsa
   ```

   A **private key** will be generated using either a default filename assigned by the OCS (e.g., id_rsa) or a filename specified by the user (e.g., my_ssh_key). This filename will be stored on the computing device in an .ssh directory under the home directory (e.g., ~/.ssh/id_rsa or ~/.ssh/my_ssh_key).

   The corresponding **public key** will be generated using the same system-default or user-specified filename, but with a .pub extension added (e.g., id_rsa.pub or my_ssh_key.pub). This public key will be stored in the same location as the private key (e.g., ~/.ssh/id_rsa.pub or ~/.ssh/my_ssh_key.pub).
4. When `gxc-config-user` queries the public key for a specific user (e.g., sudouser), enter the contents of the public key (e.g., cat ~/.ssh/id_rsa.pub).

   You now should be able to SSH to your account on the OCS (e.g., sudouser@ocs) from the computing device (e.g., host1) that has your private key (e.g., ~/.ssh/id_rsa)

   Sample Output:

   ```
   gxc-config-user> setup pubkey

   Add/Install your RSA public key to login to this OCS as (sudo or
   root) user

   You can add your RSA ssh key here. If you already have your
   public key(check your id_rsa.pub) you can add it here.
   Otherwise, please generate the public key and add it here. [The
   contents of your id_rsa.pub can be pasted below]

   Enter the loginid [root] :
   ```

```
The RSA public key needs to be provided so that it can be added
to the user's authorized keys list. It has to be in the OpenSSH
RSA public key format.(We currently support only the OpenSSH
public key format). [It is generally the content of your
id_rsa.pub file]

Please enter the public key to be installed for root : ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDEPV8mfqsGViQSyYY9rbfx7anDldEnIurW
SUxQa0+tv5W7dQxf4lkJsGP47DoM45y4robG>

Key for root set

gxc-config-user>
```

3.6.1.2 Setting Up Public-Key Authentication Natively

Public-key authentication can also be set up using standard Linux commands without utilizing `gxc-config-user`.

To set up public-key authentication natively, perform the following steps on the Linux machine from which you want to access the OCS:

1. Create a public key using `ssh-keygen`:

   ```
   ssh-keygen -t rsa
   ```

2. Store the keys and the passphrase.

3. Copy the public key using `ssh-copy-id`:

   ```
   ssh-copy-id user@<ocsipaddr>
   ```

3.7 RPC Interface

A Remote Procedure Call (RPC) interface is available to directly provision the OCS. Two types of RPC interfaces are available on the OCS: DSP and SWM. Both interfaces are works in progress and, as such, provide limited availability of the API on the switch.

Following are brief descriptions of the RPC interfaces available on the OCS:

- DSP RPC
  - Interfaces directly with the DSP, bypassing the switch matrix control algorithms. The DSP RPC should be used mainly for troubleshooting and raw low-level access.

- Switch Matrix (SWM) RPC
  - This RPC interfaces with the switch matrix and provides the API for connection management.
  - This API is more high level and internally uses the mirror control algorithms for connection management.

Following is a sample RPC interface:

```
program connectRpc {
        version connectRpc_VERSION {
 /* Establish an optical connection between an input fiber and an
output fiber.  */
makeReturnVal           makeConnectionRPC ( opticalConnection )
= 1;

           /* Remove an existing optical connection.       */
 breakReturnVal          breakConnectionRPC ( opticalConnection )
= 2;

  /* Retrieve the calibrated optical power value from a
photodetector.    */
  opticalPowerReturnStruct        retrieveOpticalPowerRPC (
opticalConnection  )  = 6;

} = 1;
```

For more information on RPC, please contact marketing@calient.net.

3.8 CORBA SDK

The CORBA SDK (Software Development Kit) is also a work in progress and contains minimal support at present. Depending upon further development of CORBA clients, this functionality can be enhanced to provide support for more interfaces.

For more information on the CORBA SDK, please contact marketing@calient.net.

4 ACRONYMS

Table 3 lists acronyms used in this guide.

Table 3 – Acronyms

Acronym	Description
API	Application Program Interface
CLI	Command Line Interface
CORBA	Common Object Request Broker Architecture
CSR	Certificate Signing Request
CURL	Command line tool for getting or sending files using URL syntax
DSP	Digital Signal Processing
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
MIB	Management Information Base
NMS	Network Management System
OCS	Optical Circuit Switch
PEM	Privacy-Enhanced Mail
REST	Representational State Transfer
RPC	Remote Procedure Call
RSA	Public-key cryptosystem used for secure data transmission
SCP	Secure Copy
SDK	Software Development Kit
SDN	Software-Defined Networking
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
SWM	Switch Matrix
WebGUI	Web Graphical User Interface