BROADCOM

Automation

Tips For Using The FOS RESTConf API

27 November 2020

Contents

PREFACE

This document is not a Brocade controlled document.

This document is intended as a supplement to existing documentation for programmers who will be writing their own modules to interact directly with the API instead using PyFOS or the supplied Ansible playbooks. It is primarily comprised of tips based on practical experience.

RESOURCES

Documentation

For details, consult the Fabric OS REST API Reference Guide, available in the Brocade Storage Networking (BSN) documentation downloads section from:

www.mybroadcom.com

Education

All Brocade education is offered at no charge. Recommended courses:

- Introduction to Fabric OS Introduction (API-220) – General overview of the REST API, PyFOS and Ansible automation topics
- Fabric OS REST Implementation (REST-320) – In depth discussion on the Fabric OS REST API
- PyFOS Installation (PyFIN-220) – Detailed discussion on PyFOS installation topics
- PyFOS Zoning (PyZONE-220) – Complete coverage of the PyFOS zoning utilities
- Introduction to the Brocade RESTCONF API", API 200-WBT

Available from:

https://www.broadcom.com/support/fibre-channel-networking/education

github

https://github.com/brocade/yang	Yang models. While the Rest API User Guide is the best source of documentation, most programmers prefer to use the Yang models for syntax and brief definitions.

https://github.com/jconsoli	API driver, zone and port configuration examples, report generator, comparison report, and statistics gathering.
https://github.com/chipcopper	A good site for Anisble resources.

ENVIRONMENT

FOS Version

The API was first introduced with FOS v8.2.0. As with most new software features, there were limitations and a few defects to work around. It is highly recommended that scripting to the API begin with FOS 8.2.1c or higher.

It's not the intent of this document to articulate the differences between what features are exposed with the various FOS levels. Check the Rest API Guide for differences.

Scripting Language

Any scripting language that supports a RESTConf API can be used with FOS.

Python Notes

All sample scripts are written in Python and require Python 3.3 or higher.

An easy and clean way to isolate your Python 3.x environment is to use the built-in virtual environment:

```
python -m venv <target-dir>
```

https://www.youtube.com/watch?v=N5vscPTWKOk

To check the path:

```
import sys
print('\n'.join(sys.path))
```

If you are using the supplied samples from https://github.com/jconsoli, use lib_check.py module, in the applications folder, to determine what required libraries are needed. This module also checks the Python path as described above.

In Windows environments, the easiest thing to do is to download the source and just copy it to your Python Lib folder. Some of the libraries you may need:

openpyxl	https://pypi.org/project/openpyxl/
jdcal.py	https://pypi.org/project/jdcal/
requests	https://pypi.org/project/requests/
urllib3	https://pypi.org/project/urllib3/
chardet	https://pypi.org/project/chardet/
certify	https://pypi.org/project/certifi/
idna	https://pypi.org/project/idna/

Reminder: *If you are in a Unix environment and just copy the libraries, you will need to set the executable attribute, -x, on all modules*.

FOS CONFIGURATION

Although FOS configuration is not mandatory, most customers will want to set up a security certificate and change the default throttling setting to improve performance.

Throttling

Although there is no known reason why throttling could not effectively be disabled, testing beyond the default throttling values has been limited. Throttling is comprised of a maximum number of requests that can be handled in a certain time frame and an idle (sleep) time required when the maximum number of requests for that window has been exceeded.

Flow control is on a per chassis basis, not a logical switch basis. All parameters discussed in the table below can be set concurrently and take effect immediately but since the parameters are applied to each request at the time the request was made, these changes effectively occur with the next request.

For most environments, the only recommended change is to increase the number of requests the chassis can handle. By default switches shipped from the factory with FOS 8.2 or upgraded in the field to 8.2 are set to handle a maximum of 30 requests in a 30

second window. After some field experience, it was determined that this default value was too low so the default for switches shipped from the factory with FOS 9.x was increased to 120 requests in a 30 second window. This throttling parameter is not changed when upgrading from FOS 8.2 to 9.0.

The maximum number of requests per chassis is 2,147,483,647. As a practical matter, the processor can't process that many requests in a 30 second window so setting the maximum is essentially disabling throttling.

Example – Configure the chassis to handle the maximum number of requests in a 30 second window:

```
mgmtapp --config  -samplerequest 2147483647
```

Example – Show the current settings:

```
mgmtapp --show
```

Detail

There is no ability to read or set the throttling parameters via the API in FOS 8.2.x or 9.0.x. To read and modify them via the CLI:

Idle time	This is the length of time to wait (sleep) after receiving a 503 status, service unavailable response. It can be set from 3 - 2147483647 seconds. By default, it is set to 3 seconds. Example – change the idle time to 6 seconds: `mgmtapp --config -idletime 6` Recommendation: There is no need to change the idle time to anything other than the default of 3 seconds; however, your code should add 1 second to this time. It was discovered in testing that with just a 3 second sleep before re-driving the request that FOS nearly always returned another 503 status. The frequency of immediately getting a 503 status after re-driving the I/O diminished as this time was incremented. At 4 seconds, the re-driven request was always accepted. In `brcdapi.brcdapi_rest`, the application sleep time is defined by `_SVC_UNAVAIL_WAIT`.

Maximum number of sessions	The maximum number of RESTConf API sessions can be 3 – 10. By default, it is set to 3. This number is not effected by Network Advisor or SANnav. Example - Change the maximum number of sessions to 10: `mgmtapp --config –maxrestsession 10` Recommendation: Unless there is a need for more than 3 sessions, leave the maximum number of sessions to the default of 3.
Sample time	The FOS throttling algorithm permits a certain number of requests within a time frame window before returning 503 status. The sample time defines this window. The sample window can be set to anything between 30 and 2147483647 seconds. By default, it is set to 30 seconds. Example - set the sample time to 60 seconds. `mgmtapp --config  -sampletime 60` Recommendation: There is no known reason at this time to ever change the sample time to anything other than the default. Should throttling be required, the better parameter to adjust is the number of sample requests.
Sample request count	This is the number of requests that can be made within the sample time before a 503 status is returned. It can be set from 30 – 2,147,483,647. As a practical matter, it's not possible to complete 2,147,483,647 requests in a 30 sec window so setting the maximum effectively disables throttling. By default, it is set to 30 in FOS 8.2 and 120 in FOS 9.0. Example – set the request count to the maximum: `mgmtapp --config  -samplerequest 2147483647` Recommendation: Although there are no known reasons why the maximum couldn't be set, testing beyond the default has been limited. The recommendation therefore is to set the sample request count to the larger default. As of this publication date, 120 is the larger default. Always read the release notes before making any changes with future releases of FOS.
Determine current settings	`mgmtapp –show`

Security Certificates

Although you can read and generate security certificates via the API, you may need to use the FOS CLI to get started.

To check to see if a certificate exists:

```
seccertmgmt show -all

sshprivatekey:
  Does not Exist

ssh public keys available for users:
  None

Certificate Files:
--------------------------------------------------------------------------
Protocol  Client CA   Server CA   SW       CSR      PVT Key  Passphrase
--------------------------------------------------------------------------
FCAP      Empty       NA          Empty    Empty    Empty    Empty
RADIUS    Empty       Empty       Empty    Empty    Empty    NA
LDAP      Empty       Empty       Empty    Empty    Empty    NA
SYSLOG    Empty       Empty       Empty    Empty    Empty    NA
HTTPS     NA          Empty       Exist    Empty    Exist    NA
KAFKA     NA          Empty       NA       NA       NA       NA
ASC       NA          Empty       NA       NA       NA       NA
```

By default, the highlighted text above will be “Empty” which means an HTTP login is required. “Exist”, as in the example above, indicates that a security certificate exists.

To auto-create a certificate which can be used with a self-signed certificate:

```
seccertmgmt generate -cert https -keysize 2048 -hash sha256
```

Hung Login Sessions

When working with the API, chances are you’ll have a bug or two that crashes your script. If this happens before you logout, your login session will remain active. Since there are a limited number of application logins supported, you may need to terminate those sessions.

If you are scripting in Python, you can use try/except between login and logout to ensure a code bug doesn’t prevent a logout but this makes it a little more challenging to get exception information.

From the CLI, there are two ways to terminate login sessions:

Reboot the Switch

```
fastboot
```

This command reboots the switch. While fast, easy, and convenient, if you are developing on shared switches, execution of this command may make you unpopular with your colleagues. It will disrupt active traffic.

Manually Disable Application Sessions

First, you'll need to determine the active session keys:

```
apploginhistory --show
```

The application logins are last but the session keys are very long and not practical to retype. Issue this command with logging enabled so you can copy and paste the session key.

```
mgmtapp -terminate session_key
```

API PROTOCOL

Throttling

If you set the maximum number of requests allowed there is no need to be concerned with throttling requests. If you did not do this, the API will return a status code of 503 if the number of requests allowed for the configured time period (30 seconds by default) has been exceeded. By default, the wait time after receiving a 503 error is 3 seconds; however, it has been observed that an addition second, a total of 4 seconds if using the default, is required to ensure a 503 error is not immediately returned when re-driving the request.

Although monitoring for 503 errors and re-driving the request alleviates the need for having to be concerned with protocol timeout in your applications, be aware that once this condition occurs, all requests thereafter are effectively throttled to just one request per 4 second window.

Carefully consider the work load. Work load does not necessarily translate to CPU utilization; however, it can effect HTTP connection timeouts. This is discussed further in the "Important API Notes" section.

HTTP Connection Timeout

The RESTConf model does not have a specified standard as to how long the processing time should be before a response is expected but there is a guideline of 20 seconds. For zoning changes and most GET methods, 20 seconds is adequate. If your script is will require additional time, see sub-section “Important Work Load Considerations”, a longer time may be required.

If the timeout is too short, the Python HTTP connect lib raises an exception but the session is not terminated on the FOS switch. This means FOS will continue to process the request. If the request was to make a change, the only way to determine if the request completed successfully when the connection times out is to read the resource and compare it against expected values.

This also means the login session remains after an HTTP connection timeout. Your script must logout after a timeout. When using Python, the HTTP connect library raises an exception when this occurs so you should consider using try/except around code that used the HTTP connect library.

HTTP Status Codes

HTTP Status	Description
200	As per standards, FOS uses this status to indicate successfully execution of a request. It is also used by brcddb and brcdapi libraries to indicate success when simulating successful request responses.
102	Not used. FOS 8.2.1c processes requests until complete. It does not time the request. Requests with many embedded requests can result in a timeout before status is received by the client so it is up to the programmer to limit the work load within a single request to avoid a timeout. Although this status has not been implemented, it is discussed herein because many programmers are familiar with it and a request to implement this status has been submitted. FOS will continue to process a request after the HTTP timeout occurs. There is no way to check on the status of a previous request after a timeout occurs.
301	Reason: “Moved Permanently” is returned by FOS when an HTTPS certificate is defined by you attempted to login as HTTP.
400	This is used as a general purpose catch all in FOS for just about any error. The “Reason” and “err-msg” may provide additional information.

	When “The Fabric is busy” is in the reason field, this indicates that the switch is too busy to process the request (typically 502 in most HTTP implementations). This typically occurs when making requests immediately after enabling a switch or after power on. The `brcdapi.brcdapi_rest` library sleeps for the time specified in `_FABRIC_BUSY_WAIT` (10 seconds as of this writing). The maximum number of retries is `_MAX_RETRIES` (5 as of this writing). This wait time and maximum number of retries is excessively long but given how infrequent this situation is, no testing was done to find more efficient parameters.
403	This is used as per standards to indicated unauthorized access such as: • Login with invalid credentials • Attempt to read/write URIs the user is not authorized to access
404	Reason: “Not found”. This is synthetically generated in brcdapi.pyfos_auth.login() when the standard Python library conn.request() raises an exception. This happens when the IP address is unreachable or HTTPS was used before a certificate was generated in the switch. If you can ping the address,
408	Not used by FOS. Used by the `brcdapi` libraries when a requests times out.
500	Used by FOS, brcdapi libraries, and brcddb libraries when a programming error is encountered. When using the brcddb or brcdapi libraries, additional information and a stack trace is written to the log. Search the log for 'Exception call with msg:'
501	“Not Implemented” – Consistent with HTTP standards, this error code is used to indicate when a method or URI is not supported by FOS. The typical error response is not returned. Instead, the error code is in the status of a normal response.
503	When “Service Unavailable” is in the reason field, this indicates that too many requests were received (typically status429). When the recommended throttling settings are set, it’s highly unlikely that you will ever see this status but your code should be designed to re-drive the request after waiting the appropriate time. See “Idle time” in the Throttling section.

Request/Response Size

The maximum length of a request passed to the API cannot exceed 10 Mbytes. As a practical matter, there is no useful request or response that takes up this much space.

Security & Certificates

In FOS 8.2.x, the HTTPS certificate is empty by default. This means all access to the API is via HTTP. The default may change in FOS 9.x. Since most production environments use secure protocols, this may have been changed during the initial switch setup.

Most environments use a simple self-signed certificate which is all that is discussed in this document. Before making any configuration changes on production switches, you should discuss the security method with your organization's security team.

See HTTP status codes 404 and 301 in HTTP Status for additional information.

In the Rest API Guide or Yang models, search for:

```
brocade-security
     security-certificate-generate
     security-certificate-action
     security-certificate
```

Important Work Load Considerations

Although work load is not CPU intensive enough to warrant concerns over CPU utilization, to perform certain actions the CPU must wait on sub-systems to complete those actions. There is no difference in time to complete actions regardless of where the action is initiated from. If work load is started from multiple sources that need access to the same resources, time to completion will be elongated.

The work load discussion in this section is focused on the timeout considerations relative to work load.

Switch Configuration

With an HTTP Connect timeout of 60 seconds, timeouts usually occurred when creating logical switches with 4 ports from two or more sessions. Timeouts always occurred when trying to configure 3 switches, each with 4 ports, at the same time. Timeouts usually occurred when creating just one switch with 4 ports while there was a `supportshow` running from an SSH session. With no other workload running, logical switch creation takes about 22 seconds. The same was observed for switch deletion. Adding ports takes about 600 msec. per port.

Recommendation: To allow for a little room, the recommended HTTP connection timeout is 60 seconds when creating a logical switch. Logical switches should not be created with any ports. When adding ports to the switch, using the same HTTP connection timeout, limit the number of ports being added in a single request to 35.

MAPS Configuration

When set to 60 seconds, HTTP timeouts were occurring in development test when executing applications/maps_config.py. This module creates all the SFP rules defined in sfp_rules_rx.xlsx. When reduced to creating just 20 rules per request, the longest request took 15 seconds.

Recommendation: Since configuring MAPS policies is rare, rather than perform extensive testing to determine under what conditions the configuration of MAPS rules were taking in excess of a few seconds, the recommendation is to limit the number of rules to 20. Additional testing should be done if you are configuring MAPS policies for something other than SFPs.

Zoning

Zoning typically requires reading some basic fabric information, including the current zone database, so that some validation can be performed and so that useful information can be presented in the event of an error.

For example, to add a server to a SAN you will want to give the HBA an alias, create a zone, add the zone to a zone configuration, and activate the zone configuration.

Example:

1. Login
2. Read basic chassis data
3. Read basic fabric data
4. Read the current zoning database
    a. At this point, the library does some basic checking. In the case of an alias, the alias name must be in a valid format and there must be a valid WWN or domain, index.
5. Send the alias updates
6. Send the zone updates
7. Send the zone configuration update.
8. Active the zone configuration (sometimes, activating the new zone configuration is a step saved for a change control window and not part of this process)
9. Logout

To estimate how long individual changes will take, multiply the number of changes by 0.4 seconds and add another 2 seconds for the basic read of data and the save. For example:

10 Alias creations

+ 4	Zone creations
+ 1	Zone configuration change (add 4 zones in one step)
= 15	Total zone changes
x 0.4	Estimated time in seconds per change
6.0	Processing time in seconds for all changes
+ 2.0	Time in seconds for other processing
= 8.0	Total time in seconds to make all changes

You can send multiple zoning changes in the same request to reduce some of this time.

Port Statistics

FOS polls statistics from the ASIC every 5 seconds in Gen5 and every 2 seconds in Gen6. The statistics are stored in a memory cache. When requesting port statistics, the data is returned from the cache. It is possible to request statistics through the API faster than the cache is refreshed resulting in the same data being returned. The timestamp, `time-generated`, returned in the response to '`brocade-interface/fibrechannel-statistics`' is a timestamp of when the request was made, not a timestamp associated with the port statistics in the response. This is a known defect which will be addressed in a later release of FOS.

Zoning

All zoning transactions take place in a zoning transaction buffer on the switch where the zoning changes are being made. The zone transactions are not sent to the fabric until the zoning changes are saved. A fabric is one or more switches connected together. Once zoning transactions are saved, the zoning transaction buffer is cleared. Any time the zoning transaction buffer is not clear, zoning transactions are outstanding.

A checksum is required by the API of the existing zone database prior to make any zone changes. The checksum is validated before pushing any changes to the fabric. Although it appears to the programmer as a single operation, enabling a zone configuration in FOS is actually a two-step process. It puts the action to enable the configuration in the transaction buffer and then pushes the zoning transaction buffer to the fabric.

You should familiarize yourself with the zoning rules as discussed in the FOS Administrators Guide before attempting to script zoning changes.

Remote Media

Usually, `brocade-media/media-rdp/remote-media-xxx`, is `None` whenever there is no remote media information available. Similarly, the associated thresholds, `remote-media-xxx-alert`, are typically `None`; however, in some instances the thresholds have been observed to be 0. These issues are typical of older optics.

It has been observed in some cases that SFPs return 0 for all remote SFP data. These observations were only made when the optic in the attached device was 8G or slower (older optics) and therefore assumed to be a problem when connected to devices with older optics. Broadcom restricts support of SFPs to ensure certain quality and standards are met but Broadcom has no control the type of SFPs used in the attached equipment.

The data associated with `remote-media-current` and `remote-media-temperature` doesn't make sense. There is either a defect in the code or documentation.

General SAN Tips

FOS Rules

All FOS rules apply to requests sent to the API. The FOS Admin Guide and FOS Command Reference Guide are useful resources for determining FOS rules.

Disabled and Recently Enabled Switches

A disabled switch is not in a fabric and therefore will not appear in any fabric requests. This doesn't happen often after a switch is put into production but it's not unusual for customers to disable a switch prior to being deployed.

A switch needs to do some work before joining a fabric so it will return a busy error message for most fabric related requests. See HTTP status code 400. The length of time will depend on the fabric design but generally this is not more than a few seconds. The most common scenario to run into this error is when trying to zone a fabric immediately after enabling a switch.

The Default Switch

From the factory, the default switch is FID 128 and is always the same as the chassis WWN. Although not often, customers do on occasion create a new logical switch, make it the default switch, and then delete the FID 128 logical switch. This means the WWN of the default switch may not be the same as the chassis WWN and the default FID may not be 128. Furthermore, the customer could have changed the default FID back to 128 in which case, the WWN of the default switch will not be the same as the chassis WWN.

An OEM may re-purposes a chassis in inventory as new if it was never shipped to a customer. Although rare, don't assume that what shipped from an OEM is at the Brocade default factory settings.

Fabric IDs

FID checking can be disabled. When FID checking is enabled, only switches with the same FID are allowed to join in a fabric together. The default FID checking state is enabled (FID must match in all switches in the fabric). FID checking should never be changed in production environments. Since all logical switch partitions in a chassis must have a unique FID, turning off FID checking can be useful in lab environments where a single chassis can be carved up into multiple logical switches and then connected together to form a fabric of multiple switches within the same chassis.

It's highly unlikely that scripts need to be concerned with multiple FIDs in the same fabric; however, since most scripting will be developed in lab environments the intent of this note is to make sure programmers are aware.

License ID and Chassis ID

Prior to Gen6, the chassis WWN and license ID were always the same. The chassis WWN and license ID may not be the same with Gen6 switches.

Aliases

Alias rules:

- Names are case sensitive
- Names must begin with a letter
    - Can be followed by any combination of letters, numbers, an underscore, and a dash.
- Cannot be more than 64 characters..
- Associated with the fabric, not a specific zone configuration
- Adding, deleting, and modifying zone aliases are part of a zoning transaction
- Changing an alias that is already used in a defined zone is permitted.
    - Keep in mind that all aliases were resolved to WWNs when the zone configuration was activated. Changing a WWN associated with an alias of a defined zone that is active will not change the effective zone until that zone configuration is re-enabled.

Typically, most organizations use one alias per WWN, do not allow multiple aliases for the same WWN, and always zone by alias. Exceptions are typically storage clusters. These are common organizational rules. They are not Fibre Channel restrictions and therefore not FOS restrictions. A WWN can have any number of aliases.

Since multiple members of an alias are permitted, they are always returned from the API as a list.

Zones

The same naming rules for aliases apply to zone rules.

Although permitted, zones containing a mix of d,i and WWN members is a bad practice. This is because session based enforcement is used. Session based zoning requires the CPU to resolve the zone and therefore adds considerable time to frame handling within the fabric. When generating zone reports, this should be a warning. All other zoning enforcement is performed in hardware with programmable arrays.

Domain, index (d,i) zones are typically used for FICON (mainframe). Although rare in distributed environments, they are sometimes used to group disk mirroring ports in a single zone.

Zoning Sources & Effect on Zoning Changes

Zoning changes are made in a switches transaction buffer and are not pushed to other switches in the fabric until the changes are saved. Zoning changes not yet pushed to a fabric are considered an outstanding zoning transaction. Although only one zoning transaction can be outstanding per switch, zoning transactions can be outstanding on multiple switches in the same fabric. Therefore, there is a potential for zone changes to be overwritten when they are pushed to the fabric.

The FCS feature was designed to avoid the potential to overwrite zoning changes by designating one switch in the fabric for zoning changes. An error is returned any time zoning changes are attempting on a switch not designated for zoning changes. Although typically the principal switch, it can be any switch in the fabric. By default, FCS is not enforced.

As of FOS v8.2.1a, setting the FCS through the API was not available; however, all FOS rules are relevant to the API as well. If FCS is defined in the fabric, the API will return an error if zoning changes are attempted on a switch not designated for zoning changes.

Sources of Zone Changes

Command Line Interface (CLI)	A CLI session can be established from: • The maintenance port ○ Typically, only used for service which does not include zoning operations • A terminal telnet session • A script with a telnet session
Network Advisor and SANnav	A management system with a proprietary API
Target Driven	Target driven zoning is a fibre channel feature that allows targets to send zoning requests in-band. Target driven zones are peer zones. They cannot be modified via any other means

	and therefore not relevant to any zoning changes. Target driven zones are reported in API requests for zone information.
Rest API	Keep in mind that your script may not be the only script attempting zoning changes.

Port State, SFPs & QSFPs

When a port is enabled, all information about the SFP can be found in `brocade-media/media-rdp`.

In FOS 8.2.1b, the leaf `physical-state` was added to `brocade-interface/fibrechannel` which returns the port state as it is returned with the `portshow` command in FOS. Any physical state other than `No_Module` indicates that an SFP is present.

When an SFP is disabled, power to the SFP is turned off. A bus in the SFP allows FOS to determine if the SFP is present but none of the other information is valid. Similarly, if a port is enabled but nothing is logged in, the Rx power level will approach negative infinity.

The same is true for QSFPs; however, each individual port on a QSFP is reported in FOS as its own SFP. All information passed through the API is presented in the same manner. Based on physical form factor, it's easy for a SAN administrator to determine what four groups of SFPs belong to a single QSFP. Since the QSFP has a single serial number, programmatically using the API the easiest way to associate SFPs with a QSFP is by the serial number which is leaf `serial-number` in `brocade-media/media-rdp`.

Name Server

Although the name server is a fabric wide database, each individual switch maintains the portion of the database associated with the logins that occurred on that switch so it will be necessary to poll all switches for name server data. This is the same as how it works with the `nsshow` command.

There are some exceptions. The details of those exceptions are beyond the scope of this document. What is important for programmers to remember is that a login may appear in the name server for multiple switches so code must be able to handle the case where logins may appear on multiple switches in the same fabric.

To obtain the name server data:

`brocade-name-server/fibrechannel-name-server?vf-id=xx`

What's important in the output:

API TIPS

Skip this section if you are using the `brcddb` libraries. The primary purpose of the `brcddb` libraries is to resolve these relationships so that they are transparent to application programming.

It is not always obvious how resources are related. For example, it's often useful to know the port number where a login occurred.

Matching Switches to Physical Chassis

The response to `'brocade-switch/fibrechannel-switch'` does not return `'chassis-wwn'` but `'brocade-fabric/fabric-switch'` does contain `'chassis-wwn'`. You will need to match the WWN of the switch in question to the switch WWN in the fabric data to find the chassis WWN. There is an example of this in `brcddb.api.interface.get_chassis()`

Matching Name Server and FDMI to Physical Switch Ports

Although logins are to a fabric, only the routing information is shared throughout the fabric. Name server and FDMI is not disturbed throughout the fabric. It is stored on the individual switches where the login occurred.

The name server information includes a reference back to the physical switch port by the WWN for the switch port but the FDMI information does not. The `name`, which is effectively the key for the resource in the FOS API, is the login WWN. Except for AMP trunk ports and SIM ports (ports in debug mode), this WWN appears in `brocade-interface/fibrechannel/neighbor`, see "Port Configuration" sub-section later in this section.

There are multiple ways to reference the physical port. In FOS 9.0, additional references were added. The `brcddb` libraries, specifically `brcddb.util.util.build_login_port_map()`, builds a map of logins to physical port spinning through the `brocade-interface/fibrechannel/neighbor` data for each port and looking those WWNs up in the fabric.

Alternatively, the fibre channel address can be used. The fibre channel address is `fcid-hex` in the port data and `port-id` in the name server data. Using the fibre channel address gets around the SIM port issue but not NPIV logins. If you use the FC address, you will have to mask off the portion of the ALPA used for NPIV logins to match it to a port. The portion of the ALPA used for NPIV logins depends on the addressing mode. By default, and for nearly all open systems environments, the portion of the ALPA used for NPIV are the lower 6 bits.

NPIV - General

Remember that NPIV enabled devices will have a base login plus one or more logical WWNs logged in. In the list of `brocade-interface/fibrechannel/neighbor` WWNs and the HBA port list in the HDMI HBA, the base WWN plus all logical logins are present. Each logical login will have its own entry in the name server and HDMI for the port.

Note: *'name-server-device-type' will be 'Physical Unknown(initiator/target)' for the base login and 'NPIV Initiator' or 'NPIV Target' for the logical logins; however, if there are no logical logins, there is no way to tell a base NPIV port from another port who's device type is unknown. As a practical matter, it is rare, if ever, that an NPIV enabled device will login to a fabric whose device behind logins will be unknown.*

NPIV – Access Gateway

A switch in access gateway mode uses NPIV so all of the comments in the NPIV – General section apply. A key difference is that those WWNs do not appear in `brocade-interface/fibrechannel/neighbor` if F-Port trunking is enabled. The `brcddb` libraries do not attempt to find the physical location of these logins so the report utility will simply show "Not found" for these logins.

Special Ports (AMP & SIM)

SIM Ports

The `brocade-interface/fibrechannel/neighbor` list of the port configuration data is empty when a port is in simulation mode (SIM-Port). The simulation port does register with the name server, `brocade-name-server/fibrechannel-name-server`. The `'port-properties'` member is `'SIM Port'`.

Ports Connected to AMP

AMP units are special switches whose connections are similar to ISLs. Just as with Remote E-Ports, the `neighbor` list of the port configuration data will contain the WWN of the AMP port and they do not present FDMI data. The trunk master registers with the name server, `brocade-name-server/fibrechannel-name-server`. The `'port-properties'` member is `'I/O Analytics Port'`. The other members of the trunk register with the name server but `'port-properties'` is not present. Also, the trunk master is AE-Port in `fibrechannel/port-type` but the other members are U-Port.

Remote E-Port

The `neighbor` list of an E-Port contains the WWN of the remote switch E-Port, see `wwn` in the "Port Configuration" subsection. Remember that switches do not register with the name server or provide any FDMI data.

Working With Fabrics

Not all fabric related data is shared with all individual switches in a fabric such as name server and FDMI. Depending on scenarios beyond the scope of this document, some fabric may be shared. Since resources are gathered from specific switches, programs combining data into a single fabric view will need to accommodate duplicate information for some data and be able to combine data for other resources.

Name Server

Response for `'/rest/running/brocade-name-server/fibrechannel-name-server'`:

`port-name`	This is the WWN of the attached device port. Matches `'PortName'` in in the FOS command `'nsshow -r'` output. This is the WWN used for zoning. This is also the WWN in `neighbor`. Note that `port-name` is not returned for a port in SIM mode or connected to AMP and the corresponding port data will not have any neighbor data.
`fabric-port-name`	This is the WWN of the physical switch port.

FDMI HBA

Response for `'/rest/running/brocade-fdmi/hba'`:

`hba-port-list`	Look in sub-key `'wwn'`. This is a list of all the port WWNs on the HBA. Matches `'port-name'` in the name server and `'port-name'` in the FDMI port.

FDMI Port

Response for `'/rest/running/brocade-fdmi/port'`:

`fabric-name`	WWN of the principal fabric switch.

port-id	Fibre channel address for the login.
port-name	The login WWN. This matches port-name in the name server which will match one of the WWNs in the FDMI HBA port list as well as one of the neighbor WWNs in the fabric switch port if the port is not in SIM mode. Note that port-name is not returned for a port in SIM mode or connected to AMP and the corresponding port data will not have any neighbor data.

Port Configuration

Response for '/rest/running/brocade-interface/fibrechannel':

fcid-hex	Fibre channel address of the port. Matches the base portion of the port-id in the FDMI.
neighbor	Sub key is 'wwn' which contains a list of all the WWNs logged into this port. These WWNs match the WWNs in brocade-name-server/fibrechannel-name-server/port-name. When NPIV is enabled, the first entry is the base login.
wwn	WWN of the physical switch port. Matches 'fabric-port-name' in the name server

Creating Reports With Excel

This section has nothing to do with the API; however, a few notes about Excel are included since generating reports in Excel is very common.

Python Excel Library

The openpyxl library can be found here:

https://pypi.org/project/openpyxl/

Sheet Names & Links

As Workbooks become large, it's common to add a table of contents with links to other sheets. Since links are to sheet names with a cell reference, not the sheet index, it's important to know the sheet name rules and some not so well known Excel rules about creating hyper-links to sheets within the workbook.

Sheet Name & Link Rules:

- No more than 31 characters
- Can't contain ':' (so no WWNs in standard format). Most other special characters are not permitted either.
- The hyperlink formula doesn't work if there are spaces or dashes in the sheet name
    - A simple solution is to convert all non-alpha numeric characters and spaces to an underscore, '_'.
- Must be unique

DRIVER & SAMPLES

Overview

https://github.com/jconsoli

It is usually easier to just use the driver in `brcdapi/brcd_rest.py` directly when setting (DELETE, PUT, PATCH, POST) anything on the switch. For a single GET operation that is true as well; however, applications performing GET requests typically need to correlate data from multiple requests. This is where using the `brcddb` libraries are useful. For example, there is a simple mechanism to get a list of all servers zoned to a certain storage which is useful for automation scripts that notify all users before performing a service action on a storage port.

When using the `brcddb` library, API login should always be performed using `brcddb.interface.login()` and `brcddb.interface.get_rest()` should always be used for all GET operations. This is because:

- `login()` reads the supported modules
    - `capture.py` uses this to determine what requests to make when capturing all data
    - Other applications use this as an expedient to determine supported requests.
- `get_batch()`
    - Is a convenient way to perform a list of GET operations
    - Automatically determines if a URI requires "`?vf-id=xx`"
    - Can perform the list of GET operations against multiple FIDs or can automatically execute the GET request against all configured logical switches in a chassis

- Uses `brcddb.api.interface.get_rest()` so response data is automatically added to the appropriate objects in the `brcddb` database.
- Formats and writes error messages to the log

brcdapi

Required by all samples in `api_examples` and `applications`.

Contains the drivers that directly interface with the API. This folder should be placed in the Python system "Lib" folder.

For customers who prefer to write their own drivers, examples on how to build content can be found in `pyfos_auth` (login and logout methods) and `brcdapi_rest` (build full URIs and handle error status).

Modules

`brcdapi_rest`	Provides a single interface to the RESTConf API in FOS. Methods in this module are used to establish, modify, send requests, and terminate sessions.
	
	• Errors indicating zero length lists are converted to 0 length lists.
	• Errors for HA requests on fixed port switches are converted to 0 length lists.
	• Service unavailable - sleep 4 seconds and retry request up to 5 times
	• Fabric busy - wait 10 seconds and retry request up to 5 times
	• Service unavailable - wait 30 seconds and retry request
	• Debug mode allows for off line work. Used with GET only
	• Processes errors
	• Debug support. See:
	  ○ brcdapi_rest Debug Mode
	  ○ Local Debug

`log`	See "Using the Built In Logging".
`pyfos_auth`	Login and logout out methods.
`util`	Defines common HTTP status and messags, converts CLI to MAPS rules, and contains a table that defines what each request is capable of an the full URI.

`zone`	Builds request content for zoning operations.

Using the Built In Logging Utility

By default, a log is automatically created with a time and date stamp in the file name. Instead of using Python print statements, all examples print to the log instead and optionally prints to STD_IO.

```
import brcdapi.log as brcdapi_log

brcdapi_log.log(msg, echo_flag, force_flag)
brcdapi_log.exception(msg, echo_flag, force_flag)
```

`msg`	Message to print to the log
`echo_flag`	If True, print the `msg` to STD_IO. The default is False.
`force_flag`	If True, ignore the global suppress printing to STD_IO and, if `echo_flag` is also True, print `msg` to STD_IO.
`_LOG_ENABLED`	True by default. When True automatically creates a log file anytime this module is imported.
`set_suppress_all()`	Suppress all output to STD_IO regardless of the `echo_flag`. This is useful for programs such as Anisble where it is desirable to suppress all logging echoed to STD_IO.
`clear_suppress_all()`	Enables echo to STD_IO.
`is_prog_suppress_all()`	Returns True if echo to STD_IO is suppressed.
`log()`	Writes a time stamped message to the log
`exception()`	Same as `log()` but adds a trace stack dump to the message and flushes the log.
`close_log()`	Closes the log file.

brcdapi_rest Debug Mode

The `brcdapi_rest` is the single interface for all requests to the API except for login and logout. See `pyfos_auth` for login and logout methods.

When `verbose_debug` is `True`, a `pprint` of all data structures sent to and received from the API is added to the log and echoed to STD_IO. By default, it is `False`. All application have the ability to enable, set `True`, `verbose_debug` via the command line. All examples have the ability to set it `True` by setting `_DEBUG_VERBOSE True`.

Local Debug

This is useful for developing scripts as it alleviates the need for a physical switch. Simulated I/O is considerably faster so it is useful even when a switch is available. It is only useful for GET requests.

Search for `_DEBUG`, `_DEBUG_MODE`, and `_DEBUG_PREFIX`. When `_DEBUG_MODE` is set to 0, all data captured with GET to a JSON dump file. When setting `_DEBUG_MODE` to 1, instead of reading data from a switch, data is read back from the file and login always returns with good status.

brcddb

Required for the sample scripts in `applications`.

This folder, if needed, should be placed in you Python system Lib folder.

These libraries make up a simple hierarchical relational database with utilities to search the database.

- A simple hierarchical relational database with utilities to search the database.
- A utility to create reports.
- Zoning utility.
  - Includes a test mode so a list of zoning operations can be passed to it and validated without making any zoning changes on the switch.
  - Look for
- Methods to login and perform GET requests as described in the Overview section.

Modules

`brcddb_*`	Several methods to perform a variety of functions such as determine the best name for a switch, check for best practice violations, perform a zone analysis, etc.

`api/zone`	Accepts a list of zoning operations which can be processed one at a time or in bulk. Includes a test mode so a list of zoning operations can be passed to it and validated without making any zoning changes on the switch.
`api/interface`	Interface to `brcdapi.brcdapi_rest.get_request()`. Response data is automatically added to the appropriate objects in the `brcddb` database.
`app_data`	Data definition tables for reports, best practice, and alerts.
`apps/zone`	Performs zoning transactions on a per transaction basis or in bulk. Originally intended for use with a script written to support an Ansible Playbook. Ansible Playbooks are usually expected to have a test mode so that the Playbook can validate the actions before executing the Playbook. It morphed into a front end for `brcdapi.zone`. A key feature is the test mode that validates zoning before attempting to send zone requests to a switch.
`apps.report`	Built in Excel report generator.
`classes/*`	Each area, such as chassis, switch, and fabric have a class object. These are the Python class definitions. These objects are the core of the `brcddb` libraries.
`report/*`	Create an Excel Workbook, add pages to the Workbook, and save the Workbook. Provides a quick and simple way to create reports from lists of objects. See `search.py` and search_dev.py for examples.
`util/*`	A collection of modules containing utilitarian methods. Too many to list them all. See the documentation embedded in each module for details.

api_examples

All of these examples require the `brcdapi` library. With the exception of login_test, all of these are intended as programming examples only. Comments in the modules are more verbose than usual. Before using any module, read the "Description" section at the beginning of each module. Contains the following:

`api_get_examples`	Examples on how to make several different GET (read) requests. This is a good place to start experimenting once you are able to login.
`login_test`	This module servers two purposes: 1. It has a user interface to enter login credentials so it can be run as a stand-alone utility to validate that a login session can be established. 2. An example of how to login and logout of a switch via the API.
`port_config`	Intended as a programming example only. It does not contain a user interface. Contains examples on how to configure ports such as: name a port, clear port statistics, etc. Programmers can set global variables for login credentials and sample code to execute. Programmers are expected to copy code segments to their own modules or run with a debugger and set breakpoints to observer behavior.
`switch_config`	Similar to `port_config` but contains switch configuration examples. It also includes a user interface.
`zone_config`	Similar to `port_config` but contains zoning examples.
`Other`	It is anticipated that additional modules will be added.

applications

With the exception of `lib_check.py`, all of the applications require the `brcddb` and `brcdapi` libraries. Some of the applications are useful as is but the primary intent of the applications is to be used as examples on how to use the `brcddb` library.

All of the applications have:

- A debug flag, `_DEBUG`, to allow global variables to be used instead of command line arguments so that programmers can set breakpoints to examine the code.
- A `-d` option which enables verbose debug, set `brcdapi.brcdapi_rest.verbose_debug` `True`.
- A `-sup` option which suppresses all output to STD_OUT.
- A `-h` option. When specified, the module displays help information and exits.

The typical use of the applications is a two-step process:

1. Capture data
2. Do something with the data

Modules

`capture`	Captures data from a single chassis. Options are to automatically capture all data for all requests that support GET, capture data from a list of KPIs, of capture data required by `report.py`. A JSON dump of the `brcddb` class objects is written to a file which is read back in for use with other applications.
`cli_zone`	Using the API as a replacement for an SSH CLI session isn't useful; however, it's a good example for those familiar with CLI zoning to use as an example of how to use `brcddb.apps.zone.py`.
`compare_report`	Creates a report in Excel Workbook with all differences between the output of capture.py or combine.py.
`lib_check`	Validates Python environment and library versions.
`multi_capture`	Starts several `capture` sessions from a list of chassis to collect data from. Upon completion, executes `combine`.
`report`	User interface to the built in `brcddb.apps.report.py` utility.
`search`	A user front end that uses the `brcddb.util.search` methods for search the output of capture.py or combine.py.
`search_dev`	Does not have a command line interface. Contains programming examples of how to use the search methods in `brcddb.util.search`.
`sfp_rules_rx`	x is a rev number. An Excel Workbook with the MAPS rule definitions that were changed or added that became the defaults in FOS v9.0.
`stats_c`	Collect port statistics from a logical switch.

`stats_g`	Converts the output from `stats_c` to an Excel Workbook. Optionally adds graphs.

Getting Started

Step 1: Install the Libraries

There is no need to do a PIP install on either `brcdapi` or `brcddb`. Simply copy the source to wherever you keep libraries. Remember to set the executable, `-x`, attribute if you are using a Unix environment. This is not necessary in DOS environments.

Step 2: Copy Sample Code

Create a work folder for script development and put the `api_get` and applications folders here.

Step 3: Python Environment Validation

This module is in the `applications` folder.

```
python lib_check.py
```

No parameters. This utility checks the python executable path, the library search paths, and the version numbers of all required libraries.

Step 4: Login and logout

This module is in the `api_examples` folder. Make sure you can login and logout via the API by executing the following:

```
python login_test.py -h
```

The `-h` option causes the utility to return the details on how to enter the login credentials.

Step 5: Do Something

A good module to start with is `api_get_examples.py` because there are only GET (read) requests.

The next step depends on what you want to do. For making changes to the switch, see api_examples.

When using the `brcddb` libraries, a good place to start is to capture data from a chassis and run a report. Executing the following from the command line will return the specific parameters needed:

```
python capture.py –h
python report.py –h
```

When running capture for the first time, keep it simple by not specifying the `–c` or `–fid` options.

When running the report for the first time, keep it simple by not specifying the `–iocp`, `–cr`, or `–ca` options. The parameter for `–sfp` should be `sfp_rules_r8.xlsx`.