Title: Brocade zone CLI generator
Author: Ian Gray
Date: 27 June 2020
Version 2

Introduction

The Brocade zone generator is a set of three utility programs to aid the setting up of a dual fabric Fibrechannel SAN which utilises Brocade switches. It is primarily targeted at new SAN configurations where there are no pre-existing configurations in place. The programs should work with any Brocade based switch. Many hardware vendors (e.g. HPE, Dell, IBM) OEM their switches from Brocade and rename them but the programs should work with them – these programs were developed on HPE badged devices.

A basic concept of this process is that a consistent naming convention is required in order that future management of the switches is made easier. The naming convention used in these programs is based on a three-level structure of node name, primary interface name, secondary interface name. Whilst some help is provided in identifying physical nodes and WWPNs it is the user's responsibility to develop a suitable naming strategy. The examples may provide some suggestions.

Note that these programs are an aide to setting up the SAN and are not a complete automation of the process.

The programs assume that the resulting zoning will be WWID based and not port based. It is assumed that the configuration will be single initiator to multiple targets. Any variation on this assumption can be controlled via the manual configuration and naming stages.

The programs work best where all hardware is in place and connected to the Brocade SAN switches and powered on to the point where the World Wide Port Names (WWPNs) are detected by the switches. Installed operating systems and storage configurations are not required by these programs.

Prerequisites

1. A Python run time/compiler environment is required. The programs were developed with Python version 3.8.2 in a Windows 10 environment. The Python environment can be downloaded from https://www.python.org/.

2. The following packages are required to be installed once the Python environment is installed. On Windows the PIP utility can be used from a command line to download and install these packages automatically.

```
pip install pyperclip
pip install wheel
pip install openpyxl
```

3. The Python code is free and unwarranted. Should the user wish to modify the code they are free to do so. For those new to Python the following environments are suggested.

For serious development the PyCharm IDE was used in the development of these programs.
https://www.jetbrains.com/pycharm/

For ease of use and quick fixes the Geany editor was used.
https://www.geany.org/

Overview

For optimal configuration of Brocade switches a set of zones is required to be established for each fabric configuration. A configuration consists of multiple zone definitions which in turn are based on alias names established for each connected device WWPN. The final output from these programs is a set of CLI commands to aid the generation of a configuration.

aliCreate commands are generated to assign a name to a WWPN.

zoneCreate and zoneAdd commands are generated to assign alias' to a zone on the basis of one initiator interface may have many target interfaces.

cfgCreate and cfgAdd commands are generated to create a configuration. Note that cfgClear, cfgDisable and cfgSave commands are also generated.

It is assumed that copy/paste will be the dominant method for deploying the commands into a switch/fabric from terminal emulation.

Note that the programs are provided as both .py (source code) and .pyc (compiled bytecode) files. The Python runtime will accept either.

The programs provided are:

SwitchAnal.py

Takes the information captured from a switch NSSHOW command and uses the information to generate a CSV file that can be used to develop a configuration by adding naming information.

Bfabric.py

Takes the manually updated spreadsheet CSV file from SwitchAnal and generates the relevant CLI commands to configure the fabrics.

WWIDfmt.py

Is a utility to aide manual input of WWPNs. Operating systems display WWPNs in a number of formats which are inconsistent with the format required by the Brocade CLI. This program scans as input string for hex characters and formats them into the required Brocade format putting the result into the paste buffer.

Examples

Two sets of example files are provided in the folders Sw3000Example and Sw6000Example. These are taken from HP branded switches. The Sw3000 example is based on a dual fabric two switch configuration. Sw6000 data is based on a dual fabric four switch configuration.

In each example:

nsshow.txt is the manually updated captured data from the switch nsshow commands.

fabanal.csv is the output from the SwitchAnal program

fabanal_named.xlsx is the manually updated csv file with values inserted in columns A, B and C and saved as a .XLSX file.

The 6 .txt files are the generated output from the Bfabric program containing the CLI commands to define the fabrics.

Program Descriptions

SwichAnal.py

By capturing the output from the Brocade CLI command “nsshow” on each switch in the fabrics and combining this data into a single file (see file nsshow.txt) it is possible to capture a great deal of useful information about the configuration as aide to generating the configuration. Note that manual intervention is required in order to insert the correct fabric identifier into the files. In the nsshow.txt example the strings “Fabric A:” and “Fabric B:” (case sensitive) are manually inserted. The “A” and “B” identifiers are propagated into the output of this program to identify the correct fabric definition commands. The Sw6000Example example is from a two fabric/four switch configuration and thus there are four “Fabric” identifier lines.

The nsshow command is a switch local command and required that the command be run on every switch in the fabric before creating a single input file into the program. Note that the switch command nsallshow is not suitable for input into the program and therefore it is not possible to collect data for a complete fabric from a single switch. Note also that the nsshow command technique is only valid if the switch can identify the connected device on each port. This equipment will need to be cabled and powered on before meaningful data can be collected.

The program looks for the strings “Fabric A:” and “Fabric B:” (case sensitive) at the beginning of the block of information for the NSSHOW commands. Typically, these lines will be manually inserted into the file immediately before the left curly bracket “{“ that starts each block of relevant information. This is used to identify which fabric the following lines are to be generated for and form the basis of column D of the output CSV file. Multiple “Fabric” statements are allowed so one can precede the information for each switch in the SAN. The fabric values obtained from a “Fabric” definition string remains a constant until changed by a subsequent “Fabric” definition.

Note that each line of output is triggered by the string “Device link speed” which is common to both 3000 and 6000 series switches. Additional data may be present beyond this “tag” which is not used and is ignored.

The output from the SwitchAnal program is a file called fabanal.csv (see example). The data in this CSV file is ordered only by the order in which the information was discovered in the input file. Only the Fabric, Target/Initiator (I/T column) and WWPN columns (D to F) are used from the derived input. Columns G onwards are there to aide identification of the end device to help with the naming information that needs to be added to columns A-C.

Note that Node and I/f columns (A, B) are mandatory for naming purposes. The Subif (column C) is optional. The sub interface can be useful for systems such as storage arrays where multiple interfaces are associated with multiple nodes on the system.

As the manually inserted information in the Node, I/f and Subif columns is used as the basis of the alias names on the switches it is recommended that short but meaningful names used e.g. use “P1” to indicate “Port-1”. Alias names are created using the values in columns A, B, C concatenated with underscores.

Note that the order of the data into the Bfabric program is not important but the program will assume that there is a header row. This leaves the spreadsheet as a tool for further analysis at the users’ discretion.

Bfabric.py

The Bfabric program accepts as input the output from the SwitchAnal program (or alternatively a file manually created in the same format) and generates files of CLI commands that can be used to configure the fabrics.

Input can either be in .xlsx format or in .CSV format. See fabanal_named.xlsx files as examples of the input.

Before generating the alias, zone and config commands the program checks the input file for duplicate WWPNs and duplicate names. The program terminates if duplicates are found.

There is an option to check for misconfigured high availability cabling. This process is based on the premise that adjoining interfaces will be on alternative fabrics and that, where multiple interfaces are to be found on a device, they have sequential WWPNs. For example – a disk controller with 4 ports would be cabled such that ports 1 and 3 were attached to fabric A and ports 2 and 4 are attached to fabric B. With a consistent naming scheme, the data can be sorted on the three name columns and the WWPN column. This produces a list where the fabric indicators alternate between A and B. If two A fabrics or two B fabrics appear in sequential rows then a cabling issue is indicated.

Selecting “N” to the question “Do you want to flag possible cable fabric/issues: (Y,N,?)” will bypass this test.

Note that the program also reports the number of records written for each fabric. If values differ e.g.

    47 alias records were written for fabric A
    45 alias records were written for fabric B

then this may also indicate a problem if it is expected that dual fabric access is the norm for all devices in the fabrics.

Taking the information provided in the XLSX or CSV file six output files are produced – three each for the two fabrics.

These files are named

Afab_ali.txt	! Alias definitions fabric A
Afab_zon.txt	! Zone definitions fabric A
Afab_cfg.txt	! Switch configuration definitions fabric A
Bfab_ali.txt	! Alias definitions fabric B
Bfab_zon.txt	! Zone definitions fabric B
Bfab_cfg.txt	! Switch configuration definitions fabric B

Connect to a switch on the relevant fabric and execute the commands by copy/paste or your preferred method of entering the data.

WWIDfmt.py

This program is a format converter for FC WWIDs. It is useful where manually obtained values from a device are to be inserted into the Bfabric input spreadsheet data. Primarily intended for Brocade formats it will return strings in a number of alternative formats which can be used via copy/paste.

Different manufactures display and accept WWPNs in different formats. The WWIDfmt program takes an input string from the user and selects characters from the string which are hexadecimal values (0-9, a-f). It will automatically put the Brocade format string into the paste buffer.

Example:

```
Colon separated strings automatically sent to the clipboard

Enter WWID to format: a145df34093423ab
a1:45:df:34:09:34:23:ab
a145df34093423ab
A145DF34093423AB
a145-df34-0934-23ab

Enter WWID to format:
```