FreeZTP

A zero-touch provisioning system built for Cisco Catalyst switches.

VERSION

The version of FreeZTP documented here is: **v0.4.0 Beta**

TABLE OF CONTENTS

WHAT IS FREEZTP

FreeZTP is a dynamic TFTP server built to automatically configure Cisco Catalyst switches upon first boot (Zero-Touch Provisioning). FreeZTP does this using the 'AutoInstall' feature built into Cisco IOS and automatically enabled by default. FreeZTP configures switches with individual, templatized configurations based upon the unique ID of the switch (usually the serial number).

REQUIREMENTS

OS: **CentOS7**

Interpreter: **Python 2.7.5+**

TERMINOLOGY

Due to the unique nature of how FreeZTP works and performs discovery of switches, there are a few

terms you will need to know to understand the application.

- **Template**
  - FreeZTP relies on the Jinja2 templating standard to take a common Cisco IOS configuration and templatize it: creating variables (with the `{{ i_am_a_variable }}` syntax) in the template where unique values can be inserted for a specific switch upon configuration push.
  - FreeZTP uses two different template types: the 'initial-template', and the custom named final templates. The initial-template is used to set the switch up for discovery, the final templates are used to push the final configuration once the discovery is complete and the switch has been identified (this will make more sense in the **ZTP Process** section).
- **Keystore**
  - The counterpart to the template (specifically: the final templates) is the keystore. The keystore is the part of the ZTP configuration which holds the unique configuration values for specific switches (or for an array of switches). The keystore provides those values for the merge of the final-template once the switch has been identified by the discovery process.
  - **Keystore ID**
    - A Keystore ID is the named identifier for a specific store which holds a set of key-value pairs.
  - **Keystore Key**
    - A Keystore Key is the key side of a certain key-value pair which resides under a named Keystore ID.
  - **Keystore Value**
    - A Keystore Value is the value side of a certain key-value pair which resides under a named Keystore ID.
  - **Keystore Hierarchy**
    - The hierarchy of the Keystore works as follows: A Keystore ID can contain multiple (unique) keys, each key with a different value. The Keystore can contain multiple IDs, each with its own set of key-value pairs.
- **ID Arrays**
  - An ID Array is a method of mapping one or more real switch IDs (ie: serial numbers) to a specific keystore. Multiple real IDs can be mapped to the same Keystore ID, which comes in handy when building a configuration for a switch stack (which could take on the serial number of any of the member switches when it boots up).
  - The ID array has two pieces:
    - The **Array Name** is the name of the specific array. The Array Name must match a Keystore ID in order to pull values from that keystore.
    - The **Array ID List** is a list of real switch IDs (serial numbers) which, when searched for, will resolve to the Array Name before mapping to a Keystore ID. When configuring an IDArray in the CLI, each ID in the list is separated by a space.
- **Associations**
  - An association is a configuration element which maps a keystore to a named template. An association is required for each keystore in order to tell FreeZTP which template to use to do the merge when using certain keystore.

ZTP PROCESS

FreeZTP relies on the 'AutoInstall' function of a Cisco Catalyst switch to configure the switch upon first boot. The process followed to configure the switch is outlined below.

1. STEP 1 - POWER ON: The Catalyst switch is powered on (or rebooted) with no startup-configuration

- REQUIREMENT: *The switch should be connected (via one of its ports) to another switch on a VLAN which is ready to serve DHCP. The DHCP scope should have DHCP OPTION 66 configured with the IP address (string) of the ZTP server.*
- NOTE: *Once the operating system is loaded on the switch and it completes the boot-up process, it will start the AutoInstall process*
- **Step 1.1:** The switch will enable all of its ports as access ports for interface Vlan1.
- **Step 1.2:** The switch will enable interface (SVI) Vlan1 and begin sending out DHCP requests from interface Vlan1.
- **Step 1.3:** Once the switch receives a DHCP lease with a TFTP server option (option 66), it will send a TFTP request for a file named "**network-confg**".
- **Step 1.4:** The switch sends a TFTP request to the TFTP server in the DHCP option 66 for a file named "**network-confg**".

2. STEP 2 - INITIAL-CONFIG: An initial config is generated, sent, and loaded for switch (target) discovery

- **Step 2.1:** When the request for the "network-confg" file is received by the ZTP server, it generates the config by performing an automatic merge with the `initial-template`:
  - **Step 2.1.1:** The `{{ autohostname }}` variable in the initial-template is filled by an automatically generated hexadecimal temporary name (example: ZTP-22F1388804). This temporary name is saved in memory by the ZTP server for future reference because the switch will use this name to request a new TFTP file in a later step
  - **Step 2.1.2:** The `{{ community }}` variable is filled with the value set in the `community` configuration field
- **Step 2.2:** This merged configuration is passed to the Cisco switch as the "network-confg" file. The switch will load it into its active running-config and proceed to step **XXXXXXXXX**
  - NOTE: *You can see an example initial configuration from the ZTP server by issuing the command* `ztp request initial-merge`

3. STEP 3 - SNMP DISCOVERY: The ZTP server discovers the switch's "real ID" (ie: serial number) using SNMP

- **Step 3.1:** After the initial config file is passed to and loaded by the switch, the ZTP server initiates a SNMP discovery of the switch
  - **Step 3.1.1:** The SNMP request targets the source IP of the switch which was used to originally request the "network-confg" file in step 1.4
  - **Step 3.1.2:** The SNMP request uses the value of the `community` configuration field as the authentication community (which the switch should honor once it loads the configuration from the "network-confg" file)
  - **Step 3.1.3:** The SNMP request uses the OID from the `snmpoid` configuration field which, by

default, is the OID to obtain the serial number of the switch
    - **Step 3.1.4:** Once the SNMP query succeeds, the ZTP server maps the real ID (ie: serial number) of the discovered switch to its temporary hostname generated in step 2.1.1

4. STEP 4 - FINAL CONFIG REQUEST: The switch requests the final configuration file and the ZTP server generates it based on the ZTP configuration

- **Step 4.1:** After the switch loads the "network-confg" file into its running-config, it sends out a new TFTP request to the ZTP server for a new filename:
    - NOTE: *The file name for the new TFTP request is based upon the hostname passed to the switch in the initial ("network-confg") file (example filename: "ZTP-22F1388804-confg")*
- **Step 4.2 (FIND A KEYSTORE ID):** ZTP attempts to find a usable Keystore ID to create the final configuration:
    - **Step 4.2.1 (USING THE REAL ID):** ZTP uses the requested filename (example filename: "ZTP-22F1388804-confg") to look up the real ID (serial number) of the requesting switch (it was saved in step 2.1.1). If the real ID of the requesting switch is known, the ZTP server attempts to use that ID to find a suitable Keystore ID:
        - **Step 4.2.1.1:** The real ID of the switch is used to search through all of the Keystore IDs to see if one of them matches the real ID. If a Keystore ID matches, then the server proceeds to **XXXXXXX** with that Keystore ID.
        - **Step 4.2.1.2:** If there is no match between the real ID and a Keystore ID, then the server looks to the ID Arrays for a match. It searches through the ID list in each IDArray, once a match is found, the server resolves the real ID to the IDArray Name and re-searches the Keystore IDs for a match using the resolved IDArray name. Once a match is found, the server continues to step **XXXXXXX** with the matched Keystore ID.
    - **Step 4.2.2 (USING THE DEFAULT KEYSTORE):** If ZTP was unable to determine the ID of the switch, or the ID of the switch does not match either a Keystore ID or an ID in an ID Array, then ZTP will look to see if a default Keystore ID has been configured using the `default-keystore` setting. If a default Keystore ID is set, and that Keystore ID is actually configured with at least one key/value pair, then ZTP will continue on through the process using that Keystore ID.
        - NOTE: *If the `default-keystore` is set to `None`, or the default-keystore name does not actually exist as a configured Keystore ID, then ZTP will send a "no such file" failure message to the switch as a response to the TFTP request*
    - NOTE: *You can test the default-keystore configuration by issuing the command* `ztp request default-keystore-test`
- **Step 4.3 (FIND THE ASSOCIATED TEMPLATE):** Once a candidate Keystore ID is found, the server checks to see if an `association` exists in the ZTP configuration for the Keystore ID. It will then use the associated template to perform the Jinja2 merge of the final configuration
    - NOTE: *If no association exists for the Keystore ID, or an association exists, but the associated template name doesn't match any configured template, then ZTP will send a "no such file" failure message to the switch as a response to the TFTP request*
- **Step 4.4 (COMPLETE THE FINAL CONFIG):** After the matching keystore and an associated template have been found, the ZTP server will perform the Jinja2 merge between the template and the key/value pairs in the Keystore ID. This configuration is then passed to the switch in response to its TFTP request made in step 4.1

- NOTE: *You can see this merged configuration by issuing the command* `ztp request merge-test <keystore-id>`
- NOTE: *If you configured static IP addresses in the final-template, then the switch starts using those static IPs and can be remotely accessible via them (assuming you also included config for AAA and SSH)*
- NOTE: *The switch does not save the new configurations into its startup-config. That has to be done manually*

Ladder Diagram

SWITCH -------> Step 1.4: File "network-confg" requested ----------> **ZTP Server**

SWITCH <------- Step 2.2: Auto-generated initial config passed to switch <---------- **ZTP Server**

SWITCH <------- Step 3: ZTP server performed discovery of real ID using SNMP <------- **ZTP Server**

SWITCH -------> Step 4.1: Switch requests new file based on new hostname ----------> **ZTP Server**

SWITCH <------- Step 4.4: ZTP responds to TFTP request with final config <---------- **ZTP Server**

INSTALLATION

The installation of FreeZTP is quick and easy using the built-in installer. Make sure you are logged in as root or are able to `sudo su` to install and operate FreeZTP.

1. Install OS with appropriate IP and OS settings and update to latest patches (recommended)
   - Check out the CentOS Minimal Server - Post-Install Setup page for help with some of the post-OS-install configuration steps.
2. Download the FreeZTP repository (may require access to the GitHub ConvergeOne organization)
3. Change to the directory where the FreeZTP main code file (ztp.py) is stored: `cd freeztp`
4. Run the FreeZTP program in install mode to perform the installation: `python ztp.py install`. Make sure the machine has internet access as this process will download and install several packages for this to happen.
   - FreeZTP will perform the install of the packages and services for full operation.
   - The installation will also install a CLI helper script. You will need to logout and log back into your SSH session to activate this helper script.

COMMAND INTERFACE

The FreeZTP service runs in the background as a system service. All commands to the FreeZTP service start with `ztp` at the command line. For example: you can check the status of the background service using the command `ztp show status`, you can check the current configuration of the ZTP server with the

command `ztp show config`.

The command interface is fully featured with helpers which can be seen either by hitting ENTER with an incomplete command in the prompt, or by using the TAB key to display available options/autocomplete the command (similar to a Cisco IOS command line, but without the use of the question mark).

All commands which change the ZTP configuration use the `set` or `clear` arguments. Commands issued with the `set` argument will overwrite an existing configuration item if that item already exists. The `clear` argument allows you to remove configuration items.

The initial and final template configurations are entered as multi-line text blocks. To facilitate this, you must specify a delineation character in the `set` command. As an example, you will issue the command `ztp set template MY_TEMPLATE ^` where the carat (`^`) character is set as the delineation character. After that command is issued, you can paste in the multi-line Cisco IOS template text. Once finished, enter that delineation character (`^` in this case) on a line by itself to exit the text block entry mode.

Below is the CLI guide for FreeZTP. You can see this at the command line by entering `ztp` and hitting ENTER (after installation).

```
-------------------------------------------------------------------------------------------------
                     ARGUMENTS                                    |
-------------------------------------------------------------------------------------------------
 - run                                                            |  Run the ZTP main program in
-------------------------------------------------------------------------------------------------
 - install                                                        |  Run the ZTP installer
 - upgrade                                                        |  Run the ZTP upgrade process
-------------------------------------------------------------------------------------------------
 - show (config|run)                                              |  Show the current ZTP configu
 - show status                                                    |  Show the status of the ZTP b
 - show version                                                   |  Show the current version of
 - show log (tail) (<num_of_lines>)                               |  Show or tail the log file
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
------------------------------------------------- SETTINGS YOU PROBABLY SHOULDN'T CHANGE ---
 - set suffix <value>                                             |  Set the file name suffix use
 - set initialfilename <value>                                    |  Set the file name used by th
 - set community <value>                                          |  Set the SNMP community you w
 - set snmpoid <value>                                            |  Set the SNMP OID to use to p
 - set initial-template <end_char>                                |  Set the initial configuratio
------------------------------------------------------------ SETTINGS YOU SHOULD CHANGE ---------
 - set template <template_name> <end_char>                        |  Create/Modify a named J2 tem
 - set keystore <id/arrayname> <keyword> <value>                  |  Create a keystore entry to b
 - set idarray <arrayname> <id_#1> <id_#2> ...                    |  Create an ID array to allow
 - set association id <id/arrayname> template <template_name>     |  Associate a keystore id or a
 - set default-keystore (none|keystore-id)                        |  Set a last-resort keystore a
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
 - clear template <template_name>                                 |  Delete a named configuration
 - clear keystore <id> (all|<keyword>)                            |  Delete an individual key or
 - clear idarray <arrayname>                                      |  Delete an ID array from the
 - clear association <id/arrayname>                               |  Delete an association from t
 - clear log                                                      |  Delete the logging info from
-------------------------------------------------------------------------------------------------
```

```
 - request merge-test <id>                                          |  Perform a test jinja2 merge
 - request initial-merge                                            |  See the result of an auto-me
 - request default-keystore-test                                    |  Check that the default-keyst
-------------------------------------------------------------------------------------------------
 - service (start|stop|restart|status)                              |  Start, Stop, or Restart the
-------------------------------------------------------------------------------------------------
 - version                                                          |  Show the current version of
-------------------------------------------------------------------------------------------------
```

The following is the default configuration seen on the ZTP server after installation when doing a `ztp show config`.

```
[root@ZTP ~]# ztp show config
!
!
!
ztp set suffix -confg
ztp set initialfilename network-confg
ztp set community secretcommunity
ztp set snmpoid 1.3.6.1.2.1.47.1.1.1.1.11.1000
!
!
ztp set initial-template ^
hostname {{ autohostname }}
!
snmp-server community {{ community }} RO
!
end
^
!
#########################################################
#########################################################
!
!
!
ztp set keystore MY_DEFAULT vl1_ip_address dhcp
ztp set keystore MY_DEFAULT hostname UNKNOWN_HOST
!
ztp set keystore SERIAL100 vl1_ip_address 10.0.0.201
ztp set keystore SERIAL100 hostname SOMEDEVICE
!
ztp set keystore STACK1 vl1_netmask 255.255.255.0
ztp set keystore STACK1 vl1_ip_address 10.0.0.200
ztp set keystore STACK1 hostname CORESWITCH
!
!
ztp set idarray STACK1 SERIAL1 SERIAL2 SERIAL3
!
!
ztp set association id MY_DEFAULT template LONG_TEMPLATE
ztp set association id SERIAL100 template SHORT_TEMPLATE
ztp set association id STACK1 template LONG_TEMPLATE
!
!
ztp set default-keystore MY_DEFAULT
```

```
!
!
!
#########################################################
ztp set template SHORT_TEMPLATE ^
hostname {{ hostname }}
other
!
interface Vlan1
 ip address {{ vl1_ip_address }} 255.255.255.0
 no shut
!
end
^
!
!
!
#########################################################
ztp set template LONG_TEMPLATE ^
hostname {{ hostname }}
!
interface Vlan1
 ip address {{ vl1_ip_address }} {{ vl1_netmask }}
 no shut
!
ip domain-name test.com
!
username admin privilege 15 secret Sigma4290!
!
aaa new-model
!
!
aaa authentication login CONSOLE local
aaa authorization console
aaa authorization exec default local if-authenticated
!
crypto key generate rsa modulus 2048
!
ip ssh version 2
!
line vty 0 15
login authentication default
transport input ssh
line console 0
login authentication CONSOLE
end
^
!
!
!
#########################################################

[root@ZTP ~]#
```

UPDATES IN V0.1.0 --> V0.2.0

ADDED FEATURES:

- Support has been added for using multiple templates in ZTP simultaneously (using custom named templates). The use of multiple templates requires that associations be created between each keystore and a named template using the `set association` command. The new configuration elements are made apparent in the new default configuration.

UPDATES IN V0.2.0 --> V0.4.0

ADDED FEATURES:

- Logging has been added to enable a user to what the background ZTP service is doing or a history of what has been done.
  - The command `ztp show log` will show the full logging file contents
  - The command `ztp show log tail` will show partial contents and will output and new logs to the terminal in real time
  - The command `ztp clear log` will clear the log file of all contents
- The `default-keystore` setting has been added which allows a keystore to be used when target discovery fails or when an unknown real ID is found during discovery. The function can also be tested by using the `ztp request default-keystore-test` command

---

CONTRIBUTING

If you would like to help out by contributing code or reporting issues, please do!

Visit the GitHub page (https://github.com/convergeone/freeztp) and either report an issue or fork the project, commit some changes, and submit a pull request.