FreeZTP

A Zero-Touch Provisioning system built for Cisco Catalyst switches.

VERSION

The version of FreeZTP documented here is: **v1.0.0**

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: Tested on **CentOS 7** (Recommended), **Ubuntu 16**, and **Raspbian Stretch Lite**

Interpreter: **Python 2.7.5+**

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 below links for easy Post-Install processes for OS's supported by FreeZTP.
    - **CentOS 7:** CentOS Minimal Server - Post-Install Setup
        - Make sure to install Git for a CentOS install
            - `sudo yum install git -y`
    - **Ubuntu 16:** Ubuntu Minimal Server - Post-Install Setup
        - Make sure to install python-pip and git for Ubuntu
            - `sudo apt install -y python-pip`
            - `sudo apt-get install -y git`
    - **Raspbian:** Raspbian Minimal Server - Post-Install Setup
        - Make sure to install python-pip and git for Raspbian
            - `sudo apt install -y python-pip`
            - `sudo apt-get install -y git`
2. Download the FreeZTP repository using Git
    - `git clone https://github.com/packetsar/freeztp.git`
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: `sudo 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 completion (helper) script. You will need to logout and log back into your SSH session to activate this completion script.

GETTING STARTED

The ZTP server comes with an [almost] fully functional default configuration, ready to serve out a basic config to switches. You can view the configuration (after installation) by issuing the command `ztp show config`.

The only part missing on the configuration is an IP lease range for DHCPD. You will need to add this IP range in order to enable the DHCPD service to hand out IP addresses (see below instructions). After this range is added and the DHCPD service restarted, you can connect the ZTP server directly to the switch (on any of its ports), power on the switch, and watch it go!

1. Configure DHCPD using the ZTP commands

    - During installation, ZTP will install the DHCPD service, detect the network interfaces in Linux, and configure DHCPD scopes for each of the interfaces. The created DHCPD scopes will be inactive to serve DHCPD as they will have no addresses available to lease.

    - If you want to use the automatically generated DHCPD scope (the new switches will be on the same VLAN as one of FreeZTPs interfaces), you just need to specify a first and last address for the lease range. After configured, you will need to commit the ZTP DHCPD configuration. Committing the DHCPD configuration (`ztp request dhcpd-commit`) automatically compiles/saves the DHCP configuration and restarts the DHCPD service. Below is example of how to do this.

        ```
        ztp set dhcpd INTERFACE-ETH0 first-address 192.168.1.100
        ztp set dhcpd INTERFACE-ETH0 last-address 192.168.1.200
        #
        ztp request dhcpd-commit
        ```

    - If the switches will not be on the same VLAN, then create a new scope (you can use the existing scope configuration commands as a reference).

    - There are other basic DHCPD options included in the scope settings like `dns-servers`, `domain-name`, and `gateway` which can be set as needed. Make sure to do a `ztp request dhcpd-commit` after any changes to DHCPD configurations.

2. Start configuring switches!

    - FreeZTP comes with a default configuration which is ready to configure switches. A default-keystore is configured which will hand out a basic (non-individualized) config to a newly booted switch.
    - There are a few example templates, keystores, associations, and an ID array already in the default config which you can reference. Feel free to modify them to do what

you want, or blow them away and customize everything.
- Once you have the ZTP configuration set, restart the ZTP service (`ztp service restart`) for your changes to take effect.
- Boot up a switch (with a blank configuration) and watch it grab a DHCP address and contact ZTP for image upgrades and configurations.
    - Be patient when the switch boots up as it sometimes takes a few (1-3) minutes after the "press RETURN to get started" message to begin the AutoInstall process.
    - You can watch for action in ZTP by checking the DHCP leases (`ztp show dhcpd leases`) and watching the active logs (`ztp show log tail`).
    - You can also see the history of devices which have gone through the provisioning process using `ztp show provisioning` .

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 a configuration pull.

    - 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. You will most likely never need to change the initial-template configuration. It has a default configuration that will most likely work for you.
        - Named (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). You will definitely be changing the named templates to fit the configurations you want your switches to have.

    - **Template Example Config**

```
ztp set template SHORT_TEMPLATE ^
hostname {{ hostname }}
```

```
!
interface Vlan1
ip address {{ vl1_ip_address }} 255.255.255.0
no shut
!
end
^
```

- **Keystore**
  - The counterpart to the template (specifically: named templates) is the keystore. The keystore is the part of the ZTP configuration which holds the unique configuration values for specific switches (or for many 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.
      - ie "SOMEID" in: `ztp set keystore SOMEID hostname SOMEDEVICE`
  - **Keystore Key**
    - A Keystore Key is the key side of a certain key-value pair which resides under a named Keystore ID.
      - ie "hostname" in: `ztp set keystore SOMEID hostname SOMEDEVICE`
  - **Keystore Value**
    - A Keystore Value is the value side of a certain key-value pair which resides under a named Keystore ID.
      - ie "SOMEDEVICE" in: `ztp set keystore SOMEID hostname SOMEDEVICE`
  - **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.
  - **Keystore Example Config**

```
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
```

- **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 will assume the serial number of the stack master 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 assign its real-IDs to 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.
  - **ID Array Example Config**

```
ztp set idarray STACK1 SERIAL1 SERIAL2 SERIAL3
```

- **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.
  - **Association Example Config**

```
ztp set association id STACK1 template LONG_TEMPLATE
```

- **Others**
  - The `default-keystore` setting allows you to specify a keystore which will be matched in the event that the switch has a real-ID (ie: serial number) which is not configured in any Keystore or IDArray. It can be set to `None` if you don't want a functional default keystore.

- The `default-template` setting associates all keystores to a template when a keystore has no specific association (ie: `ztp set association id STACK1 template LONG_TEMPLATE`)
- The `imagefile` setting specifies the image file you want to have switches use for software upgrades. The image file must exist in the TFTP root directory (/etc/ztp/tftproot/ by default). The TFTP root directory is set with the configuration command `ztp set tftproot /etc/ztp/tftproot/`.
- The `image-supression` setting prevents a second attempt by a switch to download its software image. This exists because some switches may upgrade their software, reboot into the new software version, begin the AutoInstall process again and try to upgrade again. The image-supression time setting is in seconds and is 3600 seconds (1 hour) by default.
- The `delay-keystore` setting delays the internal lookup of a keystore by ZTP for the specified number of milliseconds (1000 msec by default). This prevents ZTP from checking the SNMP discovery status before it has a chance to finish discovering the switch's real-ID using the configured SNMP OIDs.

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.

The new switch should have one of its ports connected to a network (likely an upstream switch) which has the FreeZTP server accessible. The FreeZTP server can be on the same VLAN as the new switch (so it can serve up DHCP addresses directly) or on a different VLAN which has a gateway and an IP helper pointed at the FreeZTP server so it can serve up DHCP.

1. STEP 1 - POWER ON: Initial boot up and DHCP leasing

- NOTE: *Once the operating system is loaded on the switch and it completes the boot-up process (after the "press RETURN to get started" message), it will start the AutoInstall process*
- **Step 1.1:** The switch will enable all of its ports as access ports for VLAN 1.
- **Step 1.2:** The switch will enable interface (SVI) Vlan1 and begin sending out DHCP requests from interface Vlan1.
- **Step 1.3:** The switch will get a DHCP lease from the ZTP server or from a different DHCP server. The lease will contain DHCP option 150 (TFTP Server) which points at the IP of the ZTP server.
- **Step 1.4:** To upgrade, or not to upgrade:
  - IF DHCP option 125 was configured (`ztp set dhcpd SCOPENAME imagediscoveryfile-`

`option enable`) and that option is handed to the switch in its DHCP lease, then the switch will proceed to **Step 2**.
    - IF DHCP option 125 was not configured (`ztp set dhcpd SCOPENAME imagediscoveryfile-option disable`), the switch will proceed to **Step 3**.

2. STEP 2 - IOS Upgrade: The imagediscoveryfile is used to discover the IOS image file

- NOTE: *DHCP Option 125 will contain (in hex form) the name of the imagediscoveryfile setting ("freeztp_ios_upgrade" by default in the ZTP configuration) which is a fictitious file containing the name of the .bin or .tar file the switch needs to download for the upgrade.*
- **Step 2.1:** The switch will send a TFTP request to ZTP requesting the imagediscoveryfile ("freeztp_ios_upgrade" by default).
    - **Step 2.1.1:** FreeZTP will check the file download log (seen by using `ztp show downloads`) to see if the "freeztp_ios_upgrade" file has been downloaded by that IP address within the image-supression timeframe (set with `set image-supression <seconds-to-supress>`). This time-saving feature prevents some switch models from upgrading their IOS, automatically rebooting, then attempting the upgrade again after a reboot (which results in the switch downloading the IOS image a second time, just to abort the upgrade since the software is the same version).
        - If a valid suppression entry is found, then ZTP will return the "freeztp_ios_upgrade" file containing a bogus value; causing the switch to fail the IOS download and move on to configuration downloads in **Step 3**.
        - If a valid suppression entry is NOT found, ZTP will continue from **Step 2.2**.
- **Step 2.2:** FreeZTP will check its "imagefile" (`ztp set imagefile someIOSfile.bin`) setting and dynamically generate a "freeztp_ios_upgrade" file containing the name of that .bin or .tar file. This "freeztp_ios_upgrade" file is then sent to the switch to be downloaded.
- **Step 2.3:** The switch reads the file and determines what .bin or .tar file it should download as its upgrade image. Once determined, the switch sends a TFTP download request to ZTP for that .bin or .tar filename .
- **Step 2.4:** If the .bin or .tar file does not exist, the switch abandons the upgrade attempt and proceeds to **Step 3**. If the .bin or .tar file does exist, then the ZTP server allows the switch to download it with TFTP.
- **Step 2.5:** Once fully downloaded, the switch will install the image and upgrade itself. Depending on the switch model, it may or may not automatically reboot itself. After the upgrade, the switch will start the ZTP process over, but will likely bypass the IOS upgrade step due to the image-supression function in **Step 2.1.1**.

3. STEP 3 - INITIAL-CONFIG: An initial config is generated, sent, and loaded for switch (target) discovery

- **Step 3.1:** The switch will send a TFTP request for a file named "**network-confg**" to the IP address specified in the DHCP option 150 (which should be the ZTP server).
- **Step 3.2:** 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 3.2.1:** 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 it's temporary hostname to request a new TFTP file in **Step 5.1**.
  - **Step 3.2.2:** The SNMP `{{ community }}` variable is filled with the value set in the `community` configuration field
- **Step 3.3:** This merged configuration is passed to the Cisco switch as the "network-confg" file. The switch loads it into its active running-config and proceeds to **Step 5** (ZTP performs **Step 4** in the mean time).
  - NOTE: *You can see an example initial configuration from the ZTP server by issuing the command* `ztp request initial-merge`

4. STEP 4 - SNMP DISCOVERY: The ZTP server discovers the switch's "Real ID" (ie: serial number) using SNMP

- **Step 4.1:** After the initial config file is passed to and loaded by the switch, the ZTP server initiates a SNMP discovery of the switch's serial number, or "Real ID"
  - **Step 4.1.1:** The SNMP requests target the source IP of the switch which was used to originally request the "network-confg" file in **Step 3.1**
  - **Step 4.1.2:** The SNMP requests use 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 4.1.3:** The SNMP requests use the OIDs set with `ztp set snmpoid NAME <oid>`. The FreeZTP default configuration comes with a few different OIDs pre-configured for some popular switch models.
    - NOTE: *You may need to add an OID based on the switch model you are discovering. You can test the configured OIDs for returned values using the command* `ztp request snmp-test <ip_address>` *. Your switch will need to be accessible and ready to accept the ZTP configured community*
  - **Step 4.1.4:** 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 3.2.1**

5. STEP 5 - FINAL CONFIG REQUEST: The switch requests the final configuration file and the ZTP server generates it based on the ZTP configuration

- **Step 5.1:** 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 configuration file. 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") from **Step 3.2.1**.
- **Step 5.2 (FIND A KEYSTORE ID):** ZTP attempts to find a usable Keystore ID to create the final configuration:
  - **Step 5.2.1 (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 4.1.4**). 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 5.2.1.1:** 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 **Step 5.3** with that Keystore ID.
    - **Step 5.2.1.2:** 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 5.3** with the matched Keystore ID.
  - **Step 5.2.2 (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 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 5.3 (FIND THE ASSOCIATED TEMPLATE):** Once a candidate Keystore ID is found, the server checks to see if an `association` exists in the ZTP configuration to associate the Keystore ID with a template.
  - If a specific association exists for that Keystore ID, ZTP performs the Jinja2 merge of the final configuration using the associated template.
  - If a specific association does NOT exist for that Keystore ID, ZTP checks the `default-template` setting for a usable template for a Jinja2 merge. If the setting points to an existing template, ZTP performs the Jinja2 merge of the final configuration using the default template.
  - NOTE: *If no association exists for the Keystore ID, or an association exists, but the associated template name doesn't match any configured template, AND ZTP cannot*

find a usable default template, then ZTP will send a "no such file" failure message to the switch as a response to the TFTP request

- **Step 5.4 (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 5.1**.
  - 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, the switch will then start 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 2.1: File "freeztp_ios_upgrade" requested ----------> **ZTP Server**

SWITCH <----- Step 2.1.1 or 2.2: Auto-generated "freeztp_ios_upgrade" file sent to switch <-- **ZTP Server**

SWITCH -----> Step 2.3: .bin or .tar file requested ----------> **ZTP Server**

SWITCH <----- Step 2.4: .bin or .tar file sent to switch if it exists <---------- **ZTP Server**

SWITCH -----> Step 3.1: File "network-confg" requested ----------> **ZTP Server**

SWITCH <----- Step 3.3: Auto-generated initial config passed to switch <---------- **ZTP Server**

SWITCH <----- Step 4: ZTP server performs discovery of Real ID using SNMP <---------- **ZTP Server**

SWITCH -----> Step 5.1: Switch requests new file based on new hostname ----------> **ZTP Server**

SWITCH <----- Step 5.4: ZTP responds to TFTP request with final config <----------

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.

For configuration changes to take effect, you must always restart the ZTP service using the `ztp service restart` command.

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
------------------------------------------------------------------------------
 - install                                                  |  Run the ZTP insta
 - upgrade                                                  |  Run the ZTP upgra
------------------------------------------------------------------------------
 - show (config|run) (raw)                                  |  Show the current
 - show status                                              |  Show the status o
 - show version                                             |  Show the current
```

```
 - show log (tail) (<num_of_lines>)                                  |  Show or tail the
 - show downloads (live)                                             |  Show list of TFTP
 - show dhcpd leases (current|all|raw)                               |  Show DHCPD leases
 - show provisioning                                                 |  Show list of prov
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
------------------------------------------------------ SETTINGS YOU PROBABLY SHOULDN'T
 - set suffix <value>                                                |  Set the file name
 - set initialfilename <value>                                       |  Set the file name
 - set community <value>                                             |  Set the SNMP comm
 - set snmpoid <name> <value>                                        |  Set named SNMP OI
 - set initial-template <end_char>                                   |  Set the initial c
 - set tftproot <tftp_root_directory>                                |  Set the root dire
 - set imagediscoveryfile <filename>                                 |  Set the name of t
 - set file-cache-timeout <timeout>                                  |  Set the timeout f
------------------------------------------------------------ SETTINGS YOU SHOULD CHANG
 - set template <template_name> <end_char>                           |  Create/Modify a n
 - set keystore <id/arrayname> <keyword> <value>                     |  Create a keystore
 - set idarray <arrayname> <id_#1> <id_#2> ...                       |  Create an ID arra
 - set association id <id/arrayname> template <template_name>        |  Associate a keyst
 - set default-keystore (none|keystore-id)                           |  Set a last-resort
 - set default-template (none|template_name)                         |  Set a last-resort
 - set imagefile <binary_image_file_name>                            |  Set the image fil
 - set image-supression <seconds-to-supress>                         |  Set the seconds t
 - set delay-keystore <msec-to-delay>                                |  Set the milisecon
 - set dhcpd <scope-name> [parameters]                               |  Configure DHCP sc
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
 - clear snmpoid <name>                                              |  Delete an SNMP OI
 - clear template <template_name>                                    |  Delete a named co
 - clear keystore <id> (all|<keyword>)                               |  Delete an individ
 - clear idarray <arrayname>                                         |  Delete an ID arra
 - clear association <id/arrayname>                                  |  Delete an associa
 - clear dhcpd <scope-name>                                          |  Delete a DHCP sco
 - clear log                                                         |  Delete the loggin
 - clear downloads                                                   |  Delete the list o
 - clear provisioning                                                |  Delete the list o
-------------------------------------------------------------------------------------------
 - request merge-test <id>                                           |  Perform a test ji
 - request initial-merge                                             |  See the result of
 - request default-keystore-test                                     |  Check that the de
 - request snmp-test <ip-address>                                    |  Run a SNMP test u
 - request dhcp-option-125 (windows|cisco)                           |  Show the DHCP Opt
 - request dhcpd-commit                                              |  Compile the DHCP
 - request auto-dhcpd                                                |  Automatically det
 - request ipc-console                                               |  Connect to the IP
-------------------------------------------------------------------------------------------
 - service (start|stop|restart|status)                               |  Start, Stop, or R
-------------------------------------------------------------------------------------------
 - version                                                           |  Show the current
-------------------------------------------------------------------------------------------
```

The following is the default configuration seen on the ZTP server after installation when doing

a `ztp show config`.

```
#########################################################
#
#
#
ztp set suffix -confg
ztp set initialfilename network-confg
ztp set community secretcommunity
ztp set tftproot /etc/ztp/tftproot/
ztp set imagediscoveryfile freeztp_ios_upgrade
ztp set file-cache-timeout 10
ztp set snmpoid WS-C2960_SERIAL_NUMBER 1.3.6.1.2.1.47.1.1.1.1.11.1001
ztp set snmpoid WS-C3850_SERIAL_NUMBER 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 dhcpd INTERFACE-ETH0 subnet 192.168.1.0/24
ztp set dhcpd INTERFACE-ETH0 lease-time 3600
ztp set dhcpd INTERFACE-ETH0 imagediscoveryfile-option enable
ztp set dhcpd INTERFACE-ETH0 ztp-tftp-address 192.168.1.11
#
#
#
#########################################################
#
#
#
ztp set template SHORT_TEMPLATE ^
hostname {{ hostname }}
!
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
!
!{% for interface in range(1,49) %}
interface GigabitEthernet1/0/{{interface}}
 description User Port (VLAN 1)
 switchport access vlan 1
 switchport mode access
 no shutdown
!{% endfor %}
!
ip domain-name test.com
!
username admin privilege 15 secret password123
!
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
^
#
#
#
#########################################################
#
#
#
ztp set keystore DEFAULT_VALUES vl1_ip_address dhcp
ztp set keystore DEFAULT_VALUES 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 SERIAL100 template SHORT_TEMPLATE
ztp set association id STACK1 template LONG_TEMPLATE
#
#
#
ztp set default-keystore DEFAULT_VALUES
ztp set default-template LONG_TEMPLATE
ztp set imagefile cat3k_caa-universalk9.SPA.03.06.06.E.152-2.E6.bin
ztp set image-supression 3600
ztp set delay-keystore 1000
#
#
#
##########################################################
```

DHCP FUNCTIONALITY

FreeZTP installs a DHCP service during the install. This DHCP service is used to serve IP addresses to switches with the appropriate options included.

FreeZTP commands can be used to fully configure the DHCP server. Scopes and options are created using the syntax `ztp set dhcpd SCOPENAME <option> <value>`.

FreeZTP automatically configures one or more scopes for you upon installation based on the interfaces detected in Linux and what IP addresses are on them. These scopes are needed to instruct the DHCP service to listen on those interfaces for DHCP requests. If you plan to deploy switches on the same VLAN as the FreeZTP server, then you can just add a few options to the auto-generated scope so it is ready to serve IP addresses. The installation instructions show how to do this.

After any configuration changes to the DHCP scopes, you will need to run the command `ztp request dhcpd-commit` to instruct ZTP to convert the scope settings, write them to the DHCP config file, and restart the DHCP service so the new settings take effect.

You can check on the status of the DHCP server using the command `ztp show status`

If at any point you have botched up your DHCP scopes and the DHCP service will not start back up, just delete all your scopes (using `ztp clear dhcpd` commands), and run the `ztp`

`request auto-dhcpd` command. This will rerun the DHCP discovery and auto scope creation. Then you will just need to do a `ztp request dhcpd-commit` to commit the changes and restart the service.

It is possible to use an external DHCP server instead of the FreeZTP one, but you will have to manually configure option 150 (for TFTP service from FreeZTP's IP), and option 125 (to specify the IOS image discovery file). Option 150 will need to contain the IP address of the ZTP server. You can get some instructions on how to configure option 125 by running the command `ztp request dhcp-option-125 cisco` or `ztp request dhcp-option-125 windows`.

ADVANCED USAGE

- **Complex Keystores with JSON**
  - For you more advanced Jinja2 users out there who want to set complex (multi-level) values in your keystores: a value in a keystore can be a JSON entry instead of just a flat value. Below is an example of how to do this:
    - A flat keystore value entry looks like `ztp set keystore STACK1 hostname CORESWITCH`
    - A JSON value looks like `ztp set keystore STACK1 settings '{"hostname": "CORESWITCH", "ip_address": "192.168.1.1"}'`
  - You can tell if ZTP accepted the entry as a JSON entry by using `ztp show config raw` and looking to see if the value of the `settings` key is complex instead of flat.

CONTRIBUTING

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

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