

Deploy Router Using Classic ZTP

Manually deploying network devices in a large-scale environment requires skilled workers and is time consuming.

With Zero Touch Provisioning (ZTP), you can seamlessly provision thousands of network devices accurately within minutes and without any manual intervention. This can be easily defined using a configuration file or script using shell or phyton. Currently, ZTP only supports single name-server. When the DHCP server has more than one server address configured, ZTP fails to apply the server configuration.

ZTP provides multiple options, such as:

- Automatically apply specific configuration in a large-scale environment.
- Download and install specific IOS XR image.
- Install specific application package or third party applications automatically.
- Deploy containers without manual intervention.
- Upgrade or downgrade software versions effortlessly on thousands of network devices at a time

Benefits of Using ZTP

ZTP helps you manage large-scale service providers infrastructures effortlessly. Following are the added benefits of using ZTP:

- ZTP helps you to remotely provision a router anywhere in the network. Thus eliminates the need to send an expert to deploy network devices and reduces IT cost.
- Automated provisioning using ZTP can remove delay and increase accuracy and thus is cost-effective and provides better customer experience.

  By automating repeated tasks, ZTP allows network administrators to concentrate on more important stuff.
- ZTP process helps you to quickly restore service. Rather than troubleshooting an issue by hand, you can reset a system to well-known working status.

Use Cases

The following are some of the useful use cases for ZTP:

- Using ZTP to install Chef

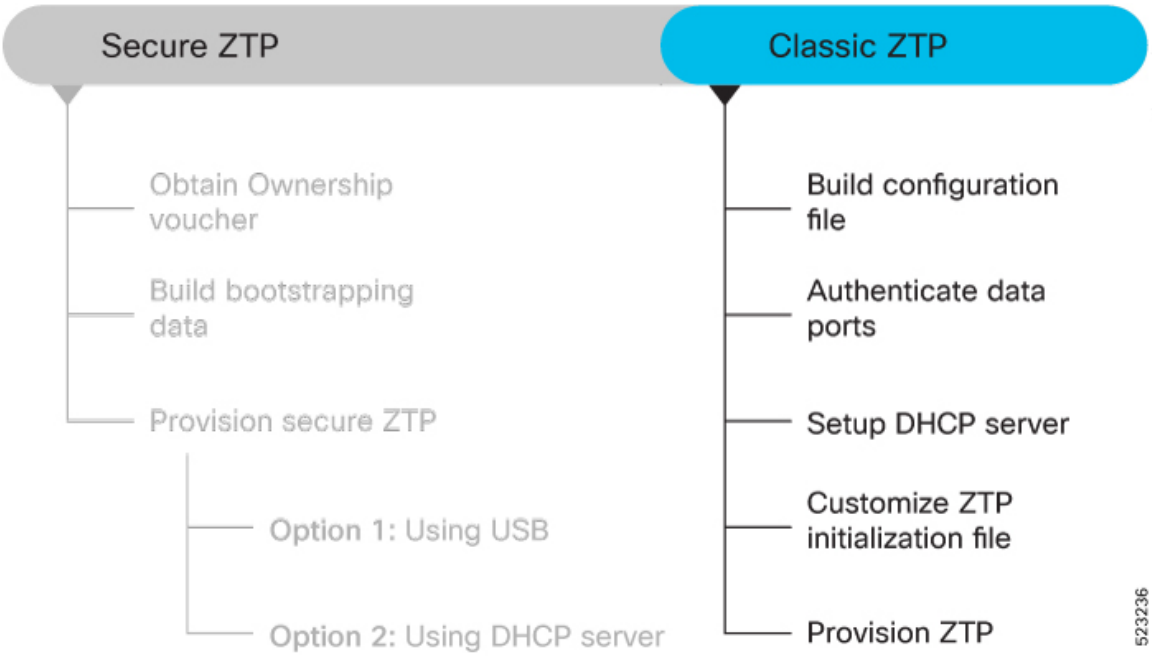
- Using ZTP to integrate IOS-XR with NSO
- Using ZTP to install Puppet

You can initiate ZTP in one of the following ways:

- **Fresh Boot**: Use this method for devices that has no pre-loaded configuration. See Getting Started with ZTP on a Fresh Boot of a Router.
- **Manual Invocation:** Use this method when you want to forcefully initiate ZTP on a fully configured device. To know the detailed steps of manual invocation of ZTP, see Manual Invocation of ZTP, on page 34.
- **ZTP Bootscript**: Use this method when you want to hard code a script to be executed on every boot.

Follow the workflow to understand the tasks involved in provisioning the router using classic ZTP.

Figure 1: Classic ZTP Workflow



This section contains the following topics:

- Deploy Router Using Classic ZTP, on page 3
- Build Configuration File, on page 4
- Limitations of temporary ZTP files, on page 20
- Authenticate Data Ports, on page 21
- Setup DHCP Server, on page 24
- Customize ZTP Initialization File, on page 27
- ZTP file download behavior with libcurl, on page 30
- Zero Touch Provisioning on a Fresh Boot of a Router, on page 30
- Manual Invocation of ZTP, on page 34

Deploy Router Using Classic ZTP

Manually deploying network devices in a large-scale environment requires skilled workers and is time consuming.

With Zero Touch Provisioning (ZTP), you can seamlessly provision thousands of network devices accurately within minutes and without any manual intervention. This can be easily defined using a configuration file or script using shell or phyton. Currently, ZTP only supports single name-server. When the DHCP server has more than one server address configured, ZTP fails to apply the server configuration.

ZTP provides multiple options, such as:

- Automatically apply specific configuration in a large-scale environment.
- Download and install specific IOS XR image.
- Install specific application package or third party applications automatically.
- Deploy containers without manual intervention.
- Upgrade or downgrade software versions effortlessly on thousands of network devices at a time

Benefits of Using ZTP

ZTP helps you manage large-scale service providers infrastructures effortlessly. Following are the added benefits of using ZTP:

- ZTP helps you to remotely provision a router anywhere in the network. Thus eliminates the need to send an expert to deploy network devices and reduces IT cost.
- Automated provisioning using ZTP can remove delay and increase accuracy and thus is cost-effective and provides better customer experience.

  By automating repeated tasks, ZTP allows network administrators to concentrate on more important stuff.
- ZTP process helps you to quickly restore service. Rather than troubleshooting an issue by hand, you can reset a system to well-known working status.

Use Cases

The following are some of the useful use cases for ZTP:

- Using ZTP to install Chef
- Using ZTP to integrate IOS-XR with NSO
- Using ZTP to install Puppet

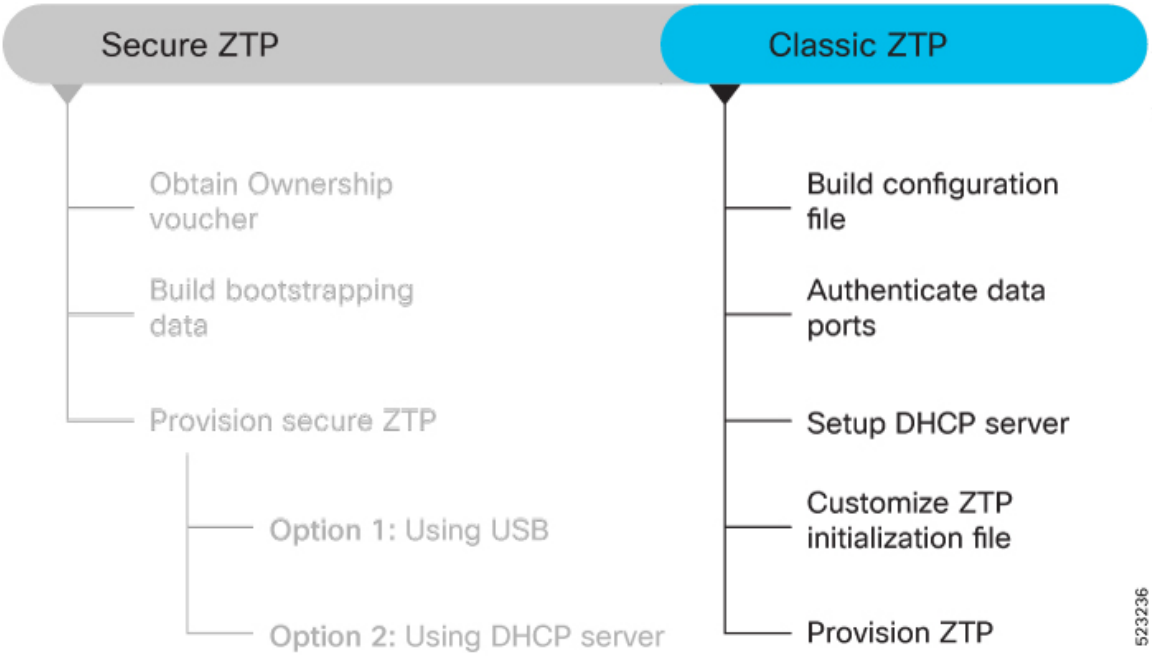
You can initiate ZTP in one of the following ways:

- **Fresh Boot**: Use this method for devices that has no pre-loaded configuration. See Getting Started with ZTP on a Fresh Boot of a Router.

- **Manual Invocation:** Use this method when you want to forcefully initiate ZTP on a fully configured device. To know the detailed steps of manual invocation of ZTP, see Manual Invocation of ZTP, on page 34.
- **ZTP Bootscript**: Use this method when you want to hard code a script to be executed on every boot.

Follow the workflow to understand the tasks involved in provisioning the router using classic ZTP.

Figure 2: Classic ZTP Workflow



This section contains the following topics:

Build Configuration File

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Build Configuration File	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Feature Name	Release Information	Feature Description
Build Configuration File	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) Based on the business need, you can use a configuration or script file to initiate the ZTP process. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Based on the business need, you can use a configuration or script file to initiate the ZTP process.

Attention When you use a USB flash drive as a source for ZTP, you cannot use the script file for provisioning. The script file is not supported in the USB fetcher. Fetcher defines which port the ZTP process should use to get the provisioning details as defined in the `ztp.ini` file.

The configuration file content starts with *!! IOS XR* and the script file content starts with *#! /bin/bash, #! /bin/sh, #! /usr/bin/python, or #! /usr/bin/python3*.

Note Update your ZTP scripts from python2 to python3, which is supported from Cisco IOS XR Release 24.1.1. You can also use python2-based ZTP scripts until Cisco IOS XR Release 25.2.1.

Once you create the configuration file, apply it to the device using the *ztp_helper* function *xrapply*.

Note We recommend that you don't execute the APIs on a router that is already provisioned. ZTP Utility APIs are designed to be executed from the ZTP script when you boot the router for the first time. The APIs perform additional operations to run the requested actions during the boot process and bring changes in the existing configuration before executing any action.

ZTP utility APIs have prerequisites which are executed in the ZTP workflow before running the ZTP utility APIs. These prerequisites help with running specific actions during the boot process and in making necessary configuration changes.

We recommend that you don't use ZTP utilities outside the scope of ZTP script. The APIs in this script use username as `ztp` or `ztp-user` in every action. The ZTP utility executed outside the scope of the ZTP script may fail as it's not executed from the ZTP workflow. This may modify the configurations on the device and affect other related operations. If the ZTP utility is executed outside the scope ZTP script, the logs display that the script is executed using username `ztp` or `ztp-user`, misleading that the script is executed from the workflow.

The following is the sample configuration file:

```
!! IOS XR
username root
group root-lr
password 0 lablab
!

hostname ios
alias exec al show alarms brief system active

interface HundredGigE 0/0/0/24
ipv4 address 10.10.10.55 255.255.255.0
no shutdown
!
```

You can also use a script file to initiate the ZTP process. This script or binary is executed in the IOS XR bash shell and can be used to interact with IOS XR CLI to configure, verify the configured state and even run EXEC commands based on the workflow that you choose. Build your ZTP script with either shell and Python.

Note ZTP does not have its own Python implementation. Instead, ZTP uses the default Python version and libraries available on the device. Python versions and supported libraries vary across Cisco IOS XR software versions and platforms (PID). Make sure that your script works correctly on the specific platform (PID) and required image version before using it in ZTP.

ZTP includes a set of CLI commands and a set of shell utilities that can be used within the user script. ZTP includes a set of shell utilities that can be sourced within the user script. The `ztp_helper.sh` is a shell script that can be sourced by the user script. This script provides simple utilities to access XR functionalities. For information on helper APIs, see the Github repository.

The following shows the sample script in python3.

```
[testenv:~]$ python3 sample_ztp_script.py
###### Debugs enabled ######

###### Change context to user specified VRF ######
```

```
###### Using Child class method, setting the root user ######
2016-12-17 04:23:24,091 - DebugZTPLogger - DEBUG - Config File content to be applied !
               username netops
               group root-lr
               group cisco-support
               secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1
               !
               end
2016-12-17 04:23:28,546 - DebugZTPLogger - DEBUG - Received exec command request: "show
configuration commit changes last 1"
2016-12-17 04:23:28,546 - DebugZTPLogger - DEBUG - Response to any expected prompt ""
Building configuration...
2016-12-17 04:23:29,329 - DebugZTPLogger - DEBUG - Exec command output is ['!! IOS XR
Configuration version = 6.2.1.21I', 'username netops', 'group root-lr', 'group cisco-support',

   'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1', '!', 'end']
2016-12-17 04:23:29,330 - DebugZTPLogger - DEBUG - Config apply through file successful,
last change = ['!! IOS XR Configuration version = 6.2.1.21I', 'username netops', 'group
root-lr', 'group cisco-support', 'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1', '!', 'end']

###### Debugs Disabled ######

###### Executing a show command ######
Building configuration...
{'output': ['!! IOS XR Configuration version = 6.2.1.21I',
        '!! Last configuration change at Sat Dec 17 04:23:25 2016 by UNKNOWN',
        '!',
        'hostname customer2',
        'username root',
        'group root-lr',
        'group cisco-support',
        'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
        '!',
        'username noc',
        'group root-lr',
        'group cisco-support',
        'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
        '!',
        'username netops',
        'group root-lr',
        'group cisco-support',
        'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
        '!',
        'username netops2',
        'group root-lr',
        'group cisco-support',
        'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
        '!',
        'username netops3',
        'group root-lr',
        'group cisco-support',
        'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
        '!',
        'cdp',
        'service cli interactive disable',
        'interface MgmtEth0/RP0/CPU0/0',
        'ipv4 address 11.11.11.59 255.255.255.0',
        '!',
        'interface TenGigE0/0/0/0/24',
        'shutdown',
        '!',
        'interface TenGigE0/0/0/0/25',
        'shutdown',
        '!',
```

```
        'router static',
        'address-family ipv4 unicast',
        '0.0.0.0/0 11.11.11.2',
        '!',
        '!',
        'end'],
        'status': 'success'}

###### Apply valid configuration using a file ######
Building configuration...
{'status': 'success', 'output': ['!! IOS XR Configuration version = 6.2.1.21I', 'hostname
customer', 'cdp', 'end']}

###### Apply valid configuration using a string ######
Building configuration...
{'output': ['!! IOS XR Configuration version = 6.2.1.21I',
        'hostname customer2',
        'end'],
'status': 'success'}

###### Apply invalid configuration using a string ######
{'output': ['!! SYNTAX/AUTHORIZATION ERRORS: This configuration failed due to',
        '!! one or more of the following reasons:',
        '!! - the entered commands do not exist,',
        '!! - the entered commands have errors in their syntax,',
        '!! - the software packages containing the commands are not active,',
```

The XML-encoded YANG configuration that follows shows various network settings including:

- Basic setup, including line configuration (TTY, VTY)
- User setup such as System Utilities
- Network configurations such as, domain service, Management IP address assignment, NETCONF, IP routing
- Protocol configurations such as, SSH, LLDP, gRPC
- Security (AAA)
- Interface settings (Interface (IF) manager)

```
Router# python3 ztp_XML_test.py
# netconf_client_ztp_lib - version 1.2 #
2021-02-22 13:53:11,587 - DebugZTPLogger - DEBUG - netconf init attempt: 1
Building configuration...
2021-02-22 13:53:18,117 - DebugZTPLogger - DEBUG - Netconf yang agent is up
###################### Netconf response: Current running configuration ######################
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <netconf xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-man-xml-ttyagent-cfg>
   /* Enables NETCONF agent over TTY*/
   <agent>
    <tty>
     <enable></enable>
    </tty>
   </agent>
  </netconf>
<lldp xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-ethernet-lldp-cfg>
 /*Enables and configures global LLDP subcommands*/
  <enable>true</enable>
</lldp>
 <ip-domain xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-ip-domain-cfg>
```

```
  /*Configures domain service related commands*/
  <vrfs>
   <vrf>
    <vrf-name>default</vrf-name>
    <name>cisco.lab</name>
    <servers>
     <server>
      <order>0</order>
      <server-address>5.38.4.246</server-address>
     </server>
    </servers>
   </vrf>
  </vrfs>
</ip-domain>
<interface-configurations xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg>
 /*Configures Interfaces and controls their activation and deactivation*/
  <interface-configuration>
   <active>act</active>
   <interface-name>HundredGigE0/0/0/14</interface-name>
   <shutdown></shutdown>
  </interface-configuration>
  <interface-configuration>
   <active>act</active>
   <interface-name>MgmtEth0/RP0/CPU0/0</interface-name>
   <ipv4-network xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-io-cfg>
    /*Configures IPv4 Interface input and output settings on the device*/
    <addresses>
     <primary>
      <address>5.38.9.29</address>
      <netmask>255.255.0.0</netmask>
     </primary>
    </addresses>
   </ipv4-network>
  </interface-configuration>
  <interface-configuration>
   <active>act</active>
   <interface-name>FourHundredGigE0/0/0/0</interface-name>
   <shutdown></shutdown>
  </interface-configuration>
</interface-configurations>
<netconf-yang xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-man-netconf-cfg>
 /*Configures Network Configuration Protocol (NETCONF) commands*/
  <agent>
   <ssh>
    <enable></enable>
   </ssh>
  </agent>
</netconf-yang>
<tty xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-tty-server-cfg>
  <tty-lines>
   <tty-line>
    <name>default</name>
    <exec>
     <timeout>
      <minutes>0</minutes>
      <seconds>0</seconds>
     </timeout>
    </exec>
    <general>
     <absolute-timeout>0</absolute-timeout>
    </general>
   </tty-line>
  </tty-lines>
</tty>
```

```
<host-names xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-cfg>
 /*Configures various system utilities related to the shell environment
of the system such as Hostname, Time zone, Prompt, Environmental variable configurations.*/

  <host-name>SF-1</host-name>
</host-names>
<grpc xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-man-ems-cfg>
  <port>57400</port>
  <no-tls></no-tls>
  <enable></enable>
</grpc>
<aaa xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-lib-cfg>
 /*Configures AAA (Authentication, Authorization, and Accounting) settings on the device*/
  <usernames xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-locald-cfg>
   <username>
    <ordering-index>0</ordering-index>
    <name>cafyauto</name>
    <usergroup-under-usernames>
     <usergroup-under-username>
      <name>root-lr</name>
     </usergroup-under-username>
     <usergroup-under-username>
      <name>cisco-support</name>
     </usergroup-under-username>
    </usergroup-under-usernames>
    <secret>
     <type>type10</type>
     <secret10>$6$iY.Zo/7E7RIG5o/.$PH1YegMZiHsiRDTxKOjKQ0i8rd4n
s2vHMHEmQrsMQrrtNTlj/gcBEQRXj3WDR8bAv0rWzz3aGdElteshHYXXR1</secret10>
    </secret>
   </username>
  </usernames>
  <accountings>
   <accounting>
    <type>commands</type>
    <listname>default</listname>
    <type-xr>start-stop</type-xr>
    <method1>local</method1>
   </accounting>
  </accountings>
</aaa>
<ssh xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-ssh-cfg>
 /*Configures the Secure Shell (SSH) settings on a device such as Encryption, Authentication,
 Session Management*/
  <server>
   <timeout>120</timeout>
   <rate-limit>600</rate-limit>
   <session-limit>110</session-limit>
   <v2></v2>
   <vrf-table>
    <vrf>
     <vrf-name>default</vrf-name>
     <enable></enable>
    </vrf>
   </vrf-table>
   <netconf>830</netconf>
   <netconf-vrf-table>
    <vrf>
     <vrf-name>default</vrf-name>
     <enable></enable>
    </vrf>
   </netconf-vrf-table>
  </server>
</ssh>
```

```
<router-static xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-ip-static-cfg>
/*Configures static IP routing on network devices*/
  <default-vrf>
   <address-family>
    <vrfipv4>
     <vrf-unicast>
      <vrf-prefixes>
       <vrf-prefix>
        <prefix>0.0.0.0</prefix>
        <prefix-length>0</prefix-length>
        <vrf-route>
         <vrf-next-hop-table>
          <vrf-next-hop-next-hop-address>
           <next-hop-address>5.38.0.1</next-hop-address>
          </vrf-next-hop-next-hop-address>
         </vrf-next-hop-table>
        </vrf-route>
       </vrf-prefix>
      </vrf-prefixes>
     </vrf-unicast>
    </vrfipv4>
   </address-family>
  </default-vrf>
</router-static>
<vty xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-tty-vty-cfg>
/*Configures virtual terminal lines (VTY lines) to access a device through SSH or TTY
protocols remotely.*/
  <vty-pools>
   <vty-pool>
    <pool-name>cafyauto</pool-name>
    <first-vty>5</first-vty>
    <last-vty>99</last-vty>
    <line-template>cafyauto</line-template>
   </vty-pool>
  </vty-pools>
</vty>
<netconf-yang xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-netconf-yang-cfg>
/*Configures the Network Configuration Protocol (NETCONF) settings and Yet Another Next
Generation (YANG) data modelling.*/
  <agent>
   <ssh/>
  </agent>
</netconf-yang>
<vty-pool xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-vty-pool-cfg>
 /*Configures virtual terminal (VTY) lines on large number of network devices*/
  <pools>
   <pool>
    <pool-name>cafyauto</pool-name>
    <first-vty-number>5</first-vty-number>
    <last-vty-number>99</last-vty-number>
    <line-template>cafyauto</line-template>
   </pool>
  </pools>
</vty-pool>
<interfaces xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-interface-cfg>
 /*Configures Interface settings such as interface, security, and performance on a device*/

  <interface>
   <interface-name>HundredGigE0/0/0/14</interface-name>
   <shutdown/>
  </interface>
  <interface>
   <interface-name>MgmtEth0/RP0/CPU0/0</interface-name>
   <ipv4>
```

```
    <addresses xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-if-ip-address-cfg>
     /*Configures IP address settings on network interfaces of a device.*/
     <address>
      <address>5.38.9.29</address>
      <netmask>255.255.0.0</netmask>
     </address>
    </addresses>
   </ipv4>
  </interface>
</interfaces>
<lldp xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-lldp-cfg/>
/*Configures the Link Layer Discovery Protocol (LLDP) settings on a network device.*/
<domain xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-domain-cfg>
 /*Configures domain settings on the device*/
  <name>cisco.lab</name>
  <name-servers>
   <name-server>
    <order>0</order>
    <address>5.38.4.246</address>
   </name-server>
  </name-servers>
</domain>
<xr-xml xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-xml-agent-cfg>
/*Configures XML agent settings such as Data formatting, Network Management, and Secure
transport layers on the router.*/
  <agent>
   <ssl/>
   <tty/>
   <enable/>
  </agent>
</xr-xml>
<netconf xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-xml-agent-cfg>
  <agent>
   <tty/>
  </agent>
</netconf>
<router xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-router-static-cfg>
 /*Configures the static routing settings on network devices*/
  <static>
   <address-family>
    <ipv4>
     <unicast>
      <prefixes>
       <prefix>
        <prefix-address>0.0.0.0</prefix-address>
        <prefix-length>0</prefix-length>
        <nexthop-addresses>
         <nexthop-address>
          <address>5.38.0.1</address>
         </nexthop-address>
        </nexthop-addresses>
       </prefix>
      </prefixes>
     </unicast>
    </ipv4>
   </address-family>
  </static>
</router>
<ssh xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-ssh-cfg>
 /*Configures the Secure Shell (SSH) settings such as secure remote access,
Encryption, and security on a network device.*/
  <timeout>120</timeout>
  <server>
   <rate-limit>600</rate-limit>
```

```
   <session-limit>110</session-limit>
   <v2/>
   <vrfs>
    <vrf>
     <vrf-name>default</vrf-name>
    </vrf>
   </vrfs>
   <netconf>
    <port>830</port>
    <vrfs>
     <vrf>
      <vrf-name>default</vrf-name>
     </vrf>
    </vrfs>
   </netconf>
  </server>
</ssh>
<grpc xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-grpc-cfg>
  /*Configures the gRPC (Google Remote Procedure Call) on a network device*/
  <port>57400</port>
  <no-tls></no-tls>
</grpc>
<hostname xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-hostname-cfg>
 /*Configures the hostname on a network device*/
  <system-network-name>SF-1</system-network-name>
</hostname>
<aaa xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-aaa-cfg>
  <usernames xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-aaa-task-user-cfg>
   /*Configures the AAA (Authentication, Authorization, and Accounting) parameters on a
network device*/
   <username>
    <ordering-index>0</ordering-index>
    <name>cafyauto</name>
    <group>
     <root-lr/>
     <cisco-support/>
    </group>
    <secret>
     <ten>$6$iY.Zo/7E7RIG5o/.$PH1YegMZiHsiRDTxKOjKQ0i8rd4ns2vHMHEmQrsMQrrtNTlj
/gcBEQRXj3WDR8bAv0rWzz3aGdElteshHYXXR1</ten>
    </secret>
   </username>
  </usernames>
  <accounting>
   <commands>
    <accounting-list>
     <list-name>default</list-name>
     <start-stop/>
     <local/>
    </accounting-list>
   </commands>
  </accounting>
 </aaa>
 <line xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-line-cfg>
   <default>
    <exec-timeout xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-line-exec-timeout-cfg>
     /*Configures the exec timeout settings on network devices for the amount of time that
 the software waits for user to input after the last key has been pressed */
     <timeout-in-minutes>0</timeout-in-minutes>
     <timeout-in-seconds>0</timeout-in-seconds>
    </exec-timeout>
    <absolute-timeout
xmlns=http://cisco.com/ns/yang/Cisco-IOS-XR-um-line-general-cfg>0</absolute-timeout>
    /*Configures line settings on network devices*/
```

```
   </default>
 </line>
 </data>
</rpc-reply>
```

Configuration Data Using JSON-encoded YANG Information

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Configuration Data Using JSON-encoded YANG Information	Release 24.1.1	We now support JSON-encoded data for YANG data instances of configuration for your router. Previously, we supported only XML-encoded data for YANG data instances.

The JSON-encoded YANG configuration that follows shows various network settings including:

- Basic setup, including line configuration (TTY, VTY)
- User setup such as System Utilities
- Network configurations such as, domain service, Management IP address assignment, NETCONF, IP routing
- Protocol configurations such as, SSH, LLDP, gRPC
- Security (AAA)
- Interface settings (Interface (IF) manager)

```
Router# python3 ztp_json_test.py
# netconf_client_ztp_lib - version 1.2 #
2021-02-22 13:53:11,587 - DebugZTPLogger - DEBUG - netconf init attempt: 1
Building configuration...
2021-02-22 13:53:18,117 - DebugZTPLogger - DEBUG - Netconf yang agent is up
###################### Netconf response: Current running configuration ######################
{
    "updates": [
        {
 /* Cisco-IOS-XR-lib-mpp-cfg: Configures library files for managing control plane services.*/


             "path":
"Cisco-IOS-XR-lib-mpp-cfg:control-plane/management-plane-protection/inband/interface-selection/all-interfaces/netconf-protocol/peer-class/peer-v4/peers/peer[address=10.10.10.1]",

            "value": {
                "address": "10.10.10.1"
            }
        },
        {
 /* Cisco-IOS-XR-lib-mpp-cfg: Configures library files for managing control plane services*/

            "path":
"Cisco-IOS-XR-lib-mpp-cfg:control-plane/management-plane-protection/inband/interface-selection/all-interfaces/netconf-protocol/peer-class/peer-v4/peer-prefixes/peer-prefix[address-prefix=10.10.10.0/24]",

            "value": {
                "address-prefix": "10.10.10.0/24"
```

```
            }
        },
        {
/* Cisco-IOS-XR-tty-management-cfg: Configures settings related to managing terminal lines
such as setting session timeouts, access control lists (ACLs)*/

            "path": "Cisco-IOS-XR-tty-server-cfg:tty/tty-lines/tty-line[name=console]",
            "value": {
                "name": "console",
                "exec": {
                    "timeout": {
                        "minutes": "0",
                        "seconds": "0"
                    }
                },
                "general": {
                    "absolute-timeout": "0"
                },
                "Cisco-IOS-XR-tty-management-cfg:connection": {
                    "session-timeout": {
                        "timeout": "0",
                        "direction": "in"
                    }
                }
            }
        },
        {
            "path": "Cisco-IOS-XR-tty-server-cfg:tty/tty-lines/tty-line[name=default]",
            "value": {
                "name": "default",
                "exec": {
                    "timeout": {
                        "minutes": "0",
                        "seconds": "0"
                    }
                },
                "general": {
                    "absolute-timeout": "0"
                },
                "Cisco-IOS-XR-tty-management-cfg:connection": {
                    "session-timeout": {
                        "timeout": "0",
                        "direction": "in"
                    }
                }
            }
        },
        {
/* Cisco-IOS-XR-aaa-lib-cfg: Configures AAA (Authentication, Authorization, and Accounting)
services.*/
            "path":
"Cisco-IOS-XR-aaa-lib-cfg:aaa/Cisco-IOS-XR-aaa-locald-cfg:usernames/username[ordering-index=0][name=cafyauto]",

            "value": {
                "ordering-index": "0",
                "name": "cafyauto",
                "usergroup-under-usernames": {
                    "usergroup-under-username": [
                        {
                            "name": "root-lr"
                        },
                        {
                            "name": "cisco-support"
                        }
```

```
                    ]
                },
                "secret": {
                    "type": "type10",
                    "secret10":
"$6$UTr4x1rhFzPoDx1.$BM7lVnhx10ipmGgVWhxb/eR4cI/x1xWjv95f1ZfCrSxMpkXMH2FxxHKOtBXaqG34A8hMoexOUfStpBwda09j/."

                }
            }
        },
        {
 /*Cisco-IOS-XR-ip-tcp-cfg: Configures library files for managing TCP/IP settings.*/
            "path":
"Cisco-IOS-XR-ip-tcp-cfg:ip/cinetd/services/telnet/vrfs/vrf[vrf-name=default]",
            "value": {
                "vrf-name": "default",
                "ipv4": {
                    "tcp": {
                        "maximum-server": "100"
                    }
                },
                "ipv6": {
                    "tcp": {
                        "maximum-server": "100"
                    }
                }
            }
        },
        {
            "path":
"Cisco-IOS-XR-ip-tcp-cfg:ip/cinetd/services/telnet/vrfs/vrf[vrf-name=genericstring]",
            "value": {
                "vrf-name": "genericstring",
                "ipv4": {
                    "tcp": {
                        "maximum-server": "16"
                    }
                },
                "ipv6": {
                    "tcp": {
                        "maximum-server": "16"
                    }
                }
            }
        },
        {
 /*Cisco-IOS-XR-linux-xlnc-cfg:Configures Linux environment (Xlnc) settings.*/
            "path":
"Cisco-IOS-XR-linux-xlnc-cfg:linux-networking/vrf-names/vrf-name[vrf-name=default]",
            "value": {
                "vrf-name": "default",
                "address-families": {
                    "address-family": [
                        {
                            "address-family": "ipv4",
                            "default-route": {
                                "software-forwarding": {}
                            },
                            "source-hint": {
                                "default-route-source-hint": {
                                    "interface-name": "MgmtEth0/RP0/CPU0/0"
                                }
                            }
                        }
```

```
                    ]
                }
            }
        },
        {
 /*Cisco-IOS-XR-ifmgr-cfg: Configures Interfaces and controls their activation and
deactivation*/
            "path":
"Cisco-IOS-XR-ifmgr-cfg:interface-configurations/interface-configuration[active=act][interface-name=FortyGigE0/0/1/1]",

            "value": {
                "active": "act",
                "interface-name": "FortyGigE0/0/1/1",
                "shutdown": {}
            }
        },
        {
            "path":
"Cisco-IOS-XR-ifmgr-cfg:interface-configurations/interface-configuration[active=act][interface-name=HundredGigE0/0/1/0]",

            "value": {
                "active": "act",
                "interface-name": "HundredGigE0/0/1/0",
                "shutdown": {}
            }
        },
        {
            "path":
"Cisco-IOS-XR-ifmgr-cfg:interface-configurations/interface-configuration[active=act][interface-name=HundredGigE0/0/1/2]",

            "value": {
                "active": "act",
                "interface-name": "HundredGigE0/0/1/2",
                "shutdown": {}
            }
        },
        {
            "path":
"Cisco-IOS-XR-ifmgr-cfg:interface-configurations/interface-configuration[active=act][interface-name=HundredGigE0/0/1/3]",

            "value": {
                "active": "act",
                "interface-name": "HundredGigE0/0/1/3",
                "shutdown": {}
            }
        },
        {
            "path":
"Cisco-IOS-XR-ifmgr-cfg:interface-configurations/interface-configuration[active=act][interface-name=MgmtEth0/RP0/CPU0/0]",

            "value": {
                "active": "act",
                "interface-name": "MgmtEth0/RP0/CPU0/0",
                "Cisco-IOS-XR-ipv4-io-cfg:ipv4-network": {
                    "addresses": {
                        "primary": {
                            "address": "10.105.247.150",
                            "netmask": "255.255.255.0"
                        }
                    }
                }
            }
        },
        {
```

```
            "path":
"Cisco-IOS-XR-ifmgr-cfg:interface-configurations/interface-configuration[active=act][interface-name=GigabitEthernet0/0/0/30]",

            "value": {
                "active": "act",
                "interface-name": "GigabitEthernet0/0/0/30",
                "shutdown": {}
            }
        },
        {
            "path":
"Cisco-IOS-XR-ifmgr-cfg:interface-configurations/interface-configuration[active=act][interface-name=GigabitEthernet0/0/0/34]",

            "value": {
                "active": "act",
                "interface-name": "GigabitEthernet0/0/0/34",
                "shutdown": {}
            }
        },
        {
            "path":
"Cisco-IOS-XR-ifmgr-cfg:interface-configurations/interface-configuration[active=act][interface-name=GigabitEthernet0/0/0/35]",

            "value": {
                "active": "act",
                "interface-name": "GigabitEthernet0/0/0/35",
                "shutdown": {}
            }
        },
        {
 /* Cisco-IOS-XR-man-ems-cfg: configures EMS (Element management System) for real time
network event detection.*/

            "path": "Cisco-IOS-XR-man-ems-cfg:grpc",
            "value": {
                "port": "57400",
                "no-tls": {},
                "enable": {}
            }
        },
        {
 /* Cisco-IOS-XR-man-netconf-cfg: Configures NETCONF (Network Configuration Protocol).*/


            "path": "Cisco-IOS-XR-man-netconf-cfg:netconf-yang/agent/ssh",
            "value": {
                "enable": {}
            }
        },
        {
 /* Cisco-IOS-XR-shellutil-cfg: Configures shell utilities in Linux.*/
            "path": "Cisco-IOS-XR-shellutil-cfg:host-names",
            "value": {
                "host-name": "150"
            }
        },
        {
            "path": "Cisco-IOS-XR-crypto-ssh-cfg:ssh/server",
            "value": {
                "v2": {}
            }
        },
        {
 /* Cisco-IOS-XR-crypto-ssh-cfg: Configures the Secure Shell (SSH) settings on a device
```

```
such as Encryption, Authentication, Session Management.*/
          "path": "Cisco-IOS-XR-crypto-ssh-cfg:ssh/server/vrf-table/vrf[vrf-name=default]",

            "value": {
                "vrf-name": "default",
                "enable": {}
            }
        },
        {
            "path":
"Cisco-IOS-XR-crypto-ssh-cfg:ssh/server/netconf-vrf-table/vrf[vrf-name=default]",
            "value": {
                "vrf-name": "default",
                "enable": {}
            }
        },
        {
            "path":
"Cisco-IOS-XR-ip-static-cfg:router-static/default-vrf/address-family/vrfipv4/vrf-unicast/vrf-prefixes/vrf-prefix[prefix=0.0.0.0][prefix-length=0]",

            "value": {
                "prefix": "0.0.0.0",
                "prefix-length": "0",
                "vrf-route": {
                    "vrf-next-hop-table": {
                        "vrf-next-hop-next-hop-address": [
                            {
                                "next-hop-address": "10.105.247.1"
                            }
                        ]
                    }
                }
            }
        },
        {
 /*Configures virtual terminal lines (VTY lines) to access a device through SSH or TTY
protocols remotely.*/

            "path": "Cisco-IOS-XR-tty-vty-cfg:vty/vty-pools/vty-pool[pool-name=default]",
            "value": {
                "pool-name": "default",
                "first-vty": "0",
                "last-vty": "50"
            }
        },
        {
 /* Cisco-IOS-XR-ethernet-lldp-cfg: Configures global LLDP subcommands.*/
            "path": "Cisco-IOS-XR-ethernet-lldp-cfg:lldp",
            "value": {
                "enable": "true",
                "enable-subintf": "true"
            }
        },
        {
 /*Cisco-IOS-XR-call-home-cfg: Configures Call Home service to contact Cisco or a designated
 server automatically for support in response to certain predefined events.*/
            "path": "Cisco-IOS-XR-call-home-cfg:call-home",
            "value": {
                "active": {},
                "contact-smart-licensing": "true"
            }
        },
        {
            "path":
```

```
"Cisco-IOS-XR-call-home-cfg:call-home/profiles/profile[profile-name=CiscoTAC-1]",
            "value": {
                "profile-name": "CiscoTAC-1",
                "active": {},
                "methods": {
                    "method": [
                        {
                            "method": "email",
                            "enable": "false"
                        },
                        {
                            "method": "http",
                            "enable": "true"
                        }
                    ]
                }
            }
        },
        {
 /* Cisco-IOS-XR-policy-repository-cfg: Configures the policy repository such as QoS,
Routing.*/
            "path":
"Cisco-IOS-XR-policy-repository-cfg:routing-policy/route-policies/route-policy[route-policy-name=PASS_ALL]",

            "value": {
                "route-policy-name": "PASS_ALL",
                "rpl-route-policy": "route-policy
PASS_ALL\n\t\t\t\t\tpass\n\t\t\t\t\tend-policy"
            }
        },
        {
 /* Cisco-IOS-XR-infra-infra-cfg: Configures infrastructure settings such as system logging,
 system clocks, banners, usernames.*/
            "path": "Cisco-IOS-XR-infra-infra-cfg:banners/banner[banner-name=login]",
            "value": {
                "banner-name": "login",
                "banner-text": "TEST"
            }
        }
    ]
}
```

Limitations of temporary ZTP files

During the ZTP (Zero Touch Provisioning) process, temporary files such as */disk0:/ztp/xr_config/interface_list* may remain on disk if the ZTP process does not exit gracefully.

Note For Cisco IOS XR software versions prior to 7.5.1, ZTP used the */disk0:/ztp/xr_config/xr_config_to_revert* file for similar tracking.

```
RP/0/RP0/CPU0:ios#run
[node0_RP0_CPU0:~]$ls /disk0\:/ztp/xr_config/
current  interface_list  xr_config_to_revert

[node0_RP0_CPU0:~]$rm /disk0\:/ztp/xr_config/interface_list
[node0_RP0_CPU0:~]$rm /disk0\:/ztp/xr_config/xr_config_to_revert
```

The ZTP process may exit abruptly under the following conditions:

- The ZTP process crashes.
- The ZTP process is manually killed.
- A reload is triggered from a ZTP script.
- The ZTP process is terminated from a script.
- A device reload or RP switchover occurs while ZTP is running.
- An RP is shut down during the ZTP process and reinserted later.

Before performing a manual upgrade, ensure that these temporary ZTP files (*/disk0:/ztp/xr_config/interface_list* and */disk0:/ztp/xr_config/xr_config_to_revert*) are deleted if they exist on disk. This prevents ZTP from bringing down configured interfaces on the next boot, even when a local username is already configured, before ZTP exits.

Authenticate Data Ports

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Authenticate Data Ports	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Feature Name	Release Information	Feature Description
Authenticate Data Ports	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) On fresh boot, the ZTP process is initiated from management ports and may switch to data ports. To validate the connection with the DHCP server, authentication is performed on data ports through DHCP option 43 for IPv4 and option 17 for IPv6. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

On fresh boot, ZTP process is initiated from management ports and may switch to data ports. To validate the connection with DHCP server, authentication is performed on data ports through DHCP option 43 for IPv4 and option 17 for IPv6. These DHCP options are defined in option space and are included within **dhcpd.conf** and **dhcpd6.conf** configuration files. You must provide following parameters for authentication while defining option space:

- Authentication code—The authentication code is either 0 or 1; where 0 indicates that authentication is not required, and 1 indicates that MD5 checksum is required.

Note If the option 43 for IPv4, and option 17 for IPv6 is disabled, the authentication fails.

- Client identifier—The client identifier must be 'exr-config' .
- MD5 checksum—This is chassis serial number. It can be obtained using **echo -n $SERIALNUMBER | md5sum | awk '{print $1}'** .

Here is the sample **dhcpd.conf** configuration. In the example below, the option space called **VendorInfo** is defined with three parameters for authentication:

```
class "vendor-classes" {
   match option vendor-class-identifier;
```

```
}

option space VendorInfo;
option VendorInfo.clientId code 1 = string;
option VendorInfo.authCode code 2 = unsigned integer 8;
option VendorInfo.md5sum code 3 = string
option vendor-specific code 43 = encapsulate VendorInfo;
subnet 10.65.2.0 netmask 255.255.255.0 {
  option subnet-mask 255.255.255.0;
  option routers 10.65.2.1;
  range 10.65.2.1 10.65.2.200;
}
host cisco-mgmt {
   hardware ethernet 00:50:60:45:67:01;
   fixed-address 10.65.2.39;
   vendor-option-space VendorInfo;
   option VendorInfo.clientId "exr-config" ;
   option VendorInfo.authCode 1;
   option VendorInfo.md5sum "aedf5c457c36390c664f5942ac1ae3829";
   option bootfile-name "http://10.65.2.1:8800/admin-cmd.sh";
}
```

Here is the sample **dhcpd6.conf** configuration file. In the example below, the option space called **VendorInfo** is defined that has code width 2 and length width 2 (as per dhcp standard for IPv6) with three parameters for authentication:

```
log-facility local7;
option dhcp6.name-servers 2001:1451:c632:1::1;
option dhcp6.domain-search "cisco.com";
dhcpv6-lease-file-name "/var/lib/dhcpd/dhcpd6.leases";
option dhcp6.info-refresh-time 21600;
option dhcp6.bootfile-url code 59 = string;
option dhcp6.user-class code 15 = string;
option space CISCO-EXR-CONFIG code width 2 length width 2;
option CISCO-EXR-CONFIG.client-identifier code 1 = string;
option CISCO-EXR-CONFIG.authCode code 2 = integer 8;
option CISCO-EXR-CONFIG.md5sum code 3 = string;
option vsio.CISCO-EXR-CONFIG code 9 = encapsulate CISCO-EXR-CONFIG;
subnet6 2001:1451:c632:1::/64{
 range6 2001:1451:c632:1::2 2001:1451:c632:1::9;
 option CISCO-EXR-CONFIG.client-identifier "exr-config";
 option CISCO-EXR-CONFIG.authCode 1;
 #valid md5
 option CISCO-EXR-CONFIG.md5sum "90fd845ac82c77f834d57a034658d0f0";
 if option dhcp6.user-class = 00:04:69:50:58:45 {
  option dhcp6.bootfile-url "http://[2001:1851:c632:1::1]/cisco-2/image.iso";
 }
 else {
   #option dhcp6.bootfile-url "http://[2001:1851:c632:1::1]/cisco-2/cisco-mini-x.iso.sh";
   option dhcp6.bootfile-url "http://[2001:1851:c632:1::1]/cisco-2/ztp.cfg";
  }
}
```

Setup DHCP Server

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Setup DHCP Server	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Setup DHCP Server	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) For ZTP to operate a valid IPv4 or IPv6 address is required and the DHCP server must send a pointer to the configuration script. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

For ZTP to operate a valid IPv4 or IPv6 address is required and the DHCP server must send a pointer to the configuration script.

The DHCP request from the router has the following DHCP options to identify itself:

- **Option 60**: “vendor-class-identifier” : Used to Identify the following four elements:
  - The type of client: For example, PXEClient
  - The architecture of The system (Arch): For example: 00009 Identify an EFI system using a x86-64 CPU
  - The Universal Network Driver Interface (UNDI):

For example 003010 (first 3 octets identify the major version and last 3 octets identify the minor version)

- The Product Identifier (PID):

- **Option 61**: “dhcp-client-identifier” : Used to identify the Serial Number of the device.
- **Option 66** : Used to request the TFTP server name.
- **Option 67**: Used request the TFTP filename.
- **Option 97**: “uuid” : Used to identify the Universally Unique Identifier a 128-bit value (not usable at this time)

Example

The following DHCP request sample provides a fixed IP address and a configuration file with the mac address of the management interface.

```
host cisco-rp0 {
   hardware ethernet e4:c7:22:be:10:ba;
   fixed-address 172.30.12.54;
   filename "http://172.30.0.22/configs/cisco-1.config";
}
```

The following DHCP request sample provides a fixed IP address and a configuration file with the mac address of the management interface along with capability to re-image the system using iPXE (exr-config "xr-config" option):

```
host cisco-rp0 {
   hardware ethernet e4:c7:22:be:10:ba;
   fixed-address 172.30.12.54;
   if exists user-class and option user-class = "iPXE" {
      filename = "http://172.30.0.22/boot.ipxe";
   } elsif exists user-class and option user-class = "exr-config" "xr-config" {
      filename = "http://172.30.0.22/scripts/cisco-rp0_ztp.sh";
   }
}
```

DHCP server identifies the device and responds with either an IOS-XR configuration file or a ZTP script as the filename option.

The DHCP server responds with the following DHCP options:

- DHCPv4 using BOOTP filename to supply script/config location.
- DHCPv4 using Option 67 (bootfile-name) to supply script/config location.
- DHCPv6 using Option 59 (OPT_BOOTFILE_URL) to supply script/config location

The following sample shows the DHCP response with bootfile-name (option 67):

```
option space cisco-vendor-id-vendor-class code width 1 length width 1;
option vendor-class.cisco-vendor-id-vendor-class code 9 = {string};

######### Network 11.11.11.0/24 #################
shared-network 11-11-11-0 {

####### Pools ##############
  subnet 11.11.11.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
```

```
 option broadcast-address 11.11.11.255;
 option routers 11.11.11.2;
 option domain-name-servers 11.11.11.2;
 option domain-name "cisco.local";
 # DDNS statements
   ddns-domainname "cisco.local.";
 # use this domain name to update A RR (forward map)
   ddns-rev-domainname "in-addr.arpa.";
   # use this domain name to update PTR RR (reverse map)

    }

######## Matching Classes ##########

  class "cisco" {
       match if (substring(option dhcp-client-identifier,0,11) = "FGE194714QS");
  }


  pool {
      allow members of "cisco";
      range 11.11.11.47 11.11.11.50;
      next-server 11.11.11.2;

      if exists user-class and option user-class = "iPXE" {
         filename="http://11.11.11.2:9090/cisco-mini-x-6.2.25.10I.iso";
      }


      if exists user-class and option user-class = "exr-config" "xr-config"
       {
          if (substring(option vendor-class.cisco-vendor-id-vendor-class,19,99)="cisco")
          {
           option bootfile-name "http://11.11.11.2:9090/scripts/exhaustive_ztp_script.py";

          }
        }


      ddns-hostname "cisco-local";
      option routers 11.11.11.2;
  }
}
```

Important In Cisco IOS XR Release 7.3.1 and earlier, the system accepts the device sending **user-class = "exr-config"**; however starting Cisco IOS XR Release 7.3.2 and later, you must use only **user-class = "xr-config"**.

In Cisco IOS XR Release 7.3.2 and later, use:

```
host cisco-rp0 {
   hardware ethernet e4:c7:22:be:10:ba;
   fixed-address 172.30.12.54;
   if exists user-class and option user-class = "iPXE" {
      filename = "http://172.30.0.22/boot.ipxe";
   } elsif exists user-class and option user-class = "xr-config" {
      filename = "http://172.30.0.22/scripts/cisco-rp0_ztp.sh";
   }
}
```

Also, when upgrading from any release that is Cisco IOS XR Release 7.3.1 or earlier to Cisco IOS XR Release 7.3.2 or later release, use the following:

```
host cisco-rp0 {
   hardware ethernet e4:c7:22:be:10:ba;
   fixed-address 172.30.12.54;
   if exists user-class and option user-class = "iPXE" {
      filename = "http://172.30.0.22/boot.ipxe";
   } elsif exists user-class and option user-class = "exr-config" {
      filename = "http://172.30.0.22/scripts/cisco-rp0_ztp.sh";
   }
}
```

Customize ZTP Initialization File

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
Saving ZTP Configuration Hashes in Router	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Saving ZTP Configuration Hashes in Router	Release 7.9.1	This feature allows you to customize **ztp.ini** file to save the ZTP configuration hashes in the **/disk0:/ztp/** location on the router. The router computes the ZTP configuration hash for all the secure ZTP configurations. The ZTP configurations hashes are useful in detecting and preventing tampering between any secure ZTP configuration and the subsequent reboot.

You can customize the following ZTP configurable options in the *ztp.ini* file:

- `ZTP`: You can enable or disable ZTP at boot using CLI or by editing the *ztp.ini* file.
- `Retry`: Set the ZTP DHCP retry mechanism: The available values are infinite and once.
- `Fetcher Priority`: Fetcher defines which port ZTP should use to get the provisioning details. By default, each port has a fetcher priority defined in the *ztp.ini* file. You can modify the default priority of the fetcher. Allowed range is from 0 to 9.

Note Lower the number higher the priority. The value 0 has the highest priority and 9 has the lowest priority.

In the following example, the Mgmt4 port has the highest priority:

```
[Fetcher Priority]
Mgmt4: 0
Mgmt6: 1
DPort4: 2
DPort6: 3
```

- `progress_bar`: Enable progress bar on the console. By default, the progress bar is disabled. To enable the progress bar, add the following entry in the `ztp.ini` file.

```
[Options]
progress_bar: True
```

- `config_check`: Saves ZTP configuration hashes in the **/disk0:/ztp/** location on the router. By default, the config check is disabled. To enable the config check, add the following entry in the`ztp.ini` file.

```
[Startup]
start:         True
retry_forever: True
config_check: True
```

You can view the ZTP hashes by using the **show ztp log** command as seen below:

```
Router# show ztp log

============================== /var/log/ztp.log ==============================

2023-03-14 12:51:29,251 53612 [Configuration] INF: Provisioning via config replace
2023-03-14 12:51:43,131 53612 [Configuration] INF: Configuration has been applied
2023-03-14 12:51:43,131 53612 [Env ] DEB: cfg::createRefOnConfigCommit: called
```

```
2023-03-14 12:51:44,218 53612 [Env ] DEB: cfg:: Generating hash for File name:
/disk0:/ztp/customer/config.inithash_tmp
2023-03-14 12:51:44,218 53612 [Env ] DEB: cfg::_generateCfgAndSaveHash:: HASH :
c7980cfc23a401bbbf296e3d49c76bf9, type : 1
2023-03-14 12:51:59,715 53612 [Env ] DEB: cfg:: Generating hash for File name:
/disk0:/ztp/customer/config.successhash_tmp
..........
2023-03-14 12:51:59,715 53612 [Env ] DEB: cfg::_generateCfgAndSaveHash:: HASH :
c7980cfc23a401bbbf296e3d49c76bf9, type : 2
2023-03-14 12:52:04,901 53612 [Env ] DEB: cfg::getRefOnSuccess :: called
..........
2023-03-14 12:52:05,403 53612 [Engine ] INF: ZAdmin, current state:active, exit
code:success
2023-03-14 12:52:05,403 53612 [Engine ] INF: ZAdmin, current state:final, exit
code:success: state changed to final
```

By default, the `ztp.ini` file is located in the `/pkg/etc/` location. To modify the ZTP configurable options, make a copy of the file in the `/disk0:/ztp/` directory and then edit the `ztp.ini` file.

To reset to the default options, delete the ztp.ini file in the `/disk0:/ztp/` directory.

Note Do not edit or delete the `ztp.ini` file in the `/pkg/etc/` location to avoid issues during installation.

The following example shows the sample of the `ztp.ini` file:

```
[Startup]
start: True
retry_forever: True

[Fetcher Priority]
Mgmt4:  1
Mgmt6:  2
DPort4: 3
DPort6: 4
```

Enable ZTP Using CLI

If you want to enable ZTP using CLI, use the **ztp enable** command.

Configuration example

```
Router#ztp enable
Fri Jul 12 16:09:02.154 UTC
Enable ZTP? [confirm] [y/n] :y
ZTP Enabled.
```

Disable ZTP Using CLI

If you want to disable ZTP using CLI, use the **ztp disable** command.

Configuration example

```
Router#ztp disable
Fri Jul 12 16:07:18.491 UTC
Disable ZTP? [confirm] [y/n] :y
ZTP Disabled.
Run ZTP enable to run ZTP again.
```

Prior to Cisco IOS XR Release 7.5.1, the ztp disable command creates a `ztp.ini` file in the `/disk0:/ztp` folder. Starting with Cisco IOS XR Release 7.5.1 and later, the ztp.ini file is created in the

`/disk0:/ztp/internal` folder. This file tracks whether ZTP is disabled, ensuring persistence of the disabled state during upgrades and downgrades.

However, the disabled state of ZTP is not persistent when reimaging is performed between pre-7.5.1 and post-7.5.1 images. This means reimaging across these release boundaries requires reconfiguration to disable ZTP again.

If ZTP was disabled:

- Before upgrading from a pre-7.5.1 image, copy `/disk0:/ztp/ztp.ini` to `/disk0:/ztp/internal`.
- Before downgrading to a pre-7.5.1 image, copy `/disk0:/ztp/internal/ztp.ini` to `/disk0:/ztp`.

ZTP file download behavior with libcurl

Cisco IOS XR ZTP uses the `libcurl` library to download provisioning files such as scripts and software images in releases 24.1.1, 24.2.1, 24.2.2, and 24.2.21. The ZTP process retrieves the download URL from DHCP options and passes it directly as a string to libcurl, which initiates the file transfer.

Key behaviors and considerations for ZTP file downloads with `libcurl` include:

- If the download URL uses HTTPS, `libcurl` automatically attempts the HTTPS connection. ZTP allows downloads from HTTPS servers without needing explicit certificate verification configuration, so HTTPS downloads succeed without extra flags or options.
- Administrators sometimes add the `-k` option to a curl command to allow insecure HTTPS connections. However, ZTP does not interpret `-k` as an option; it treats it as part of the DHCP-provided URL string.

  For example, using a DHCP-provided value like *https://server.example.com/ztp.script -k* causes ZTP to interpret `-k` as part of the path, likely resulting in a 404 error or download failure.
- Do not include command-line options such as `-k` in DHCP-provided ZTP URLs in releases 24.1.1, 24.2.1, 24.2.2, and 24.2.21. Use only a valid file URL, such as `https://server.example.com/ztp.script`.
- If needed, you can disable libcurl file downloads in the ZTP configuration file (`ztp.ini`) in releases 24.1.1, 24.2.1, 24.2.2, and 24.2.21 by adding:

  ```
  [Http]
  libcurl_http_get: False
  ```

  When `libcurl` is disabled, ZTP uses the alternative download method as defined by the framework or falls back to the Linux curl utility.

Following these guidelines ensures that ZTP reliably downloads files using libcurl without errors caused by invalid options in the URL.

Zero Touch Provisioning on a Fresh Boot of a Router

When you boot the device, the ZTP process initiates automatically if the device does not have a prior configuration.

Fresh boot using removable storage device

You can automatically provision a network device using ZTP from a removable storage device such as a USB flash drive. The following configuration types are supported on the removable storage device:

- Device-specific configuration: Located in a folder named after the device's chassis serial number. For example, the path for device-specific configuration is `/USB-path/xr-config/<serial-number>/router-cfg`. For instance, `/USB-path/xr-config/FOC2102R1D0/router-cfg` where `FOC2102R1D0` is the chassis serial number.
- Generic configuration: Located directly in the `xr-config` folder. The sample path is `/USB-path/xr-config/router-cfg`.

High-level workflow of ZTP using a USB flash drive

1. When the device boots, it checks if USB is enabled in the `ztp.ini` file. By default, the USB *fetcher* is enabled and assigned the highest priority.

   The fetcher defines which port ZTP uses to obtain provisioning details. Each port has a fetcher priority defined in the `ztp.ini` file.

2. ZTP checks for the presence of a USB flash drive on the device. If no USB drive is detected, ZTP moves to the next fetcher based on the fetcher priority in the `ztp.ini` file.

3. If a USB flash drive is available, ZTP looks for the provisioning files in the following order:

   a. `<usb-root>/xr-config/<chassis-serial>/ztp_actions.json`

   b. `<usb-root>/xr-config/ztp_actions.json`

   c. `<usb-root>/xr-config/<chassis-serial>/router-cfg`

   d. `<usb-root>/xr-config/router-cfg`

4. If a `ztp_actions.json` file is found, ZTP uses only the configuration path specified inside that JSON file.

   Sample generic configuration:

   ```
   "global": {
       "actions": {
         "config": "xr-config/common/router-cfg"
   ```

5. For device-specific configuration, ZTP looks for the `router-cfg` file inside the folder matching the device's chassis serial number within the `xr-config` folder and applies this device-specific configuration. For example, `/USB-path/xr-config/FOC2102R1D0/router-cfg`.

   Device-specific configuration:

   ```
   "FOC2102R1D0": {
       "actions": {
         "image-install": {
           "boot-image": {
             "os-version": "7.9.1",
             "path": "xr-config/FOC2102R1D0/8000-x64-7.9.1.iso"
           }
         }
       }
     }
   ```

For information about the JSON keys used above, see JSON keys and options in ztp_actions file.

6. If the device-specific configuration is not found, ZTP looks for the generic `router-cfg` file directly inside the `xr-config` folder and applies this generic configuration.
7. If the config path inside `ztp_actions.json` is missing or the file is absent, ZTP retries and does not fall back to `router-cfg`.
8. The device applies the retrieved configuration.
9. The network device is now fully provisioned and operational.

JSON keys and options in ztp_actions.json file

The supported JSON keys and options in `ztp_actions.json` include:

- Top-level keys:
  - `global`: Generic or default action block used when no device-specific block matches.
  - `<chassis-serial>`: Device-specific action block keyed by chassis serial number (e.g., `FOC2102R1D0`), which takes priority over `global`.
  - `actions`: Container for actual ZTP actions.
- Action keys under `actions`:
  - `config`: String path to the XR config file to apply. Required for normal config provisioning. If absent, ZTP retries.
  - `image-install`: Requests image upgrade/install before config apply.
    - `boot-image` sub-container includes:
      - `os-version`: Requested image version string.
      - `path`: Path to ISO/GISO image on USB.
  - `enable-secure-mode`: Boolean (`true` or `false`). When `true`, ZTP enables secure mode and restarts ZTP to continue in Secure ZTP flow. Image or config actions in the same file are ignored. If unsupported, ZTP moves to the next fetcher. If enabling secure mode fails, ZTP exits to avoid undefined state. Default is `false` if absent.

Configure ZTP using USB

Follow these steps to configure ZTP using a USB flash drive:

1. Create the configuration file. See *Build your Configuration File*.



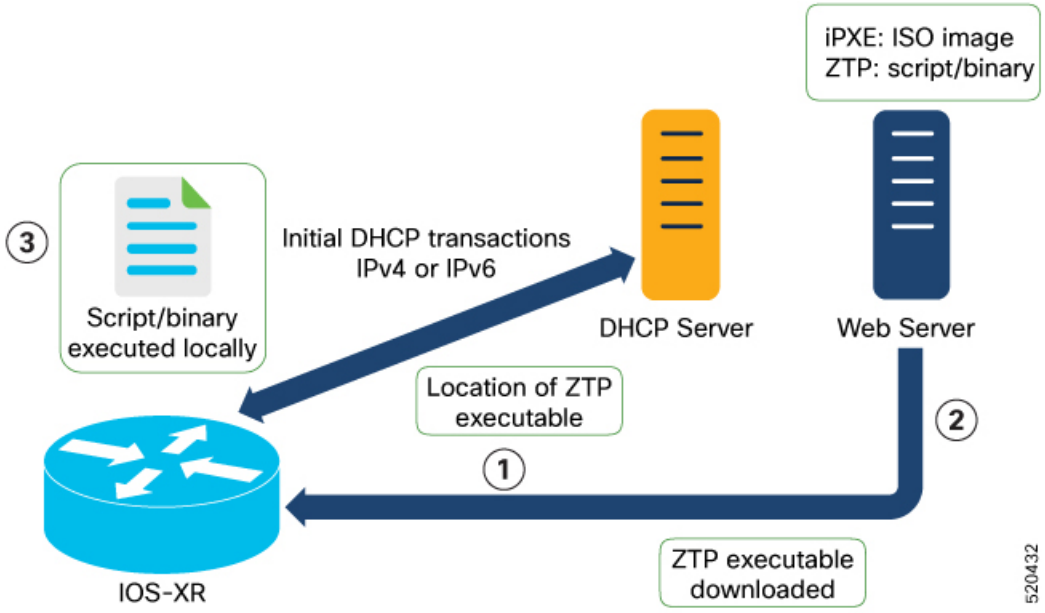
Note When you use a USB flash drive as a source for ZTP, you can't use the script file for provisioning. The script file isn't supported for USB fetcher.

2. Copy the bootstrapping data to the USB flash drive and mount it on the device.

Fresh Boot Using DHCP

When you boot the device, the ZTP process initiates automatically if the device does not have a prior configuration. During the process, the router receives the details of the configuration file from the DHCP server.

This image depicts the high-level work flow of the ZTP process:



The ZTP process initiates when you boot the network-device with an IOS-XR image. The process starts only on the device that doesn't have a prior configuration.

Here is the high-level work flow of the ZTP process for the Fresh boot:

1. ZTP sends DHCP request to fetch the ZTP configuration file or user script. To help the Bootstrap server uniquely identify the device, ZTP sends below DHCP option

   - DHCP(v4/v6) client-id=Serial Number
   - DHCPv4 option 124: Vendor, Platform, Serial-Number
   - DHCPv6 option 16: Vendor, Platform, Serial-Number

   The following is the default sequential flow of the ZTP process:

   - ZTP sends IPv4 DHCP request first on all the management port. In case there is a failure, then ZTP sends IPv6 DHCP request on all the management port.
   - ZTP sends IPv4 DHCP request first on all the data port. In case there is a failure, then ZTP sends IPv6 DHCP request on all the data port.

     The default sequential flow is defined in configuration file and you can modify the sequence using the configuration file.

2. DHCP server identifies the device and responds with DHCP response using one of the following options:

   DHCP server should be configured to respond with the DHCP options.

- DHCPv4 using BOOTP filename to supply script/config location.
- DHCPv4 using Option 67 (bootfile-name) to supply script/config location.
- DHCPv6 using Option 59 (OPT_BOOTFILE_URL) to supply script/config location

3. The network device downloads the file from the web server using the URI location that is provided in the DHCP response.
4. The device receives a configuration file or script file from the HTTP server.

Note
- If the downloaded file content starts with !! IOS XR it is considered as a configuration file.
- If the downloaded file content starts with #! /bin/bash, #! /bin/sh or #!/usr/bin/python it is considered as a script file.

5. The device applies the configuration file or executes the script or binary in the default bash shell.
6. The Network device is now up and running.

Manual Invocation of ZTP

Procedure

Step 1 Use the **ztp clean** command to remove previous state information from ZTP.

Example:

```
Router#ztp clean
         This would remove all ZTP temporary files.
         Would you like to proceed? [no]: yes
         All ZTP operation files have been removed.
         ZTP logs are present in /var/log/ztp*.log for logrotate.
         Please remove manually if needed.
         If you now wish ZTP to run again from boot, do 'configure terminal/commit replace' followed
 by reload
```

Step 2 Use the **commit replace** command to replace the entire running configuration.

Example:

```
Router#configure terminal
Router(configure)#commit replace
execute "commit replace from configuration mode"
```

Note
- ZTP triggers only if there's no configuration/username.
- The execution of ZTP can be affected by an existing configuration on interface such as DHCP or ACL.
- If the ZTP work flow involves image upgrade, the presence of configuration makes ZTP to exit after reload.

- Existing configuration can cause issue to image upgrade.
- If the existing configuration isn't compatible with the new image upgraded you'll see failures after reimage.

Step 3 Do one of the following steps.

- Use the **reload** command to initiate the ZTP process automatically during system boot on active RP.

```
Router#reload
```

- Use the **ztp initiate** command to initiate ZTP manually.

```
Router#ztp initiate
Initiating ZTP may change your configuration.
Interfaces might be brought up if they are in shutdown state
Would you like to proceed? [no]: yes
ZTP will now run in the background.
```

If all conditions are met ZTP continues with fetch, otherwise exits. The **ztp initiate** command manually initiates ZTP for testing, bypassing all ZTP checks that is followed in the reload method. You shouldn't execute **ztp initiate** on up and running testbed, this command modifies the existing configuration on the router. If the execution of **ztp initiate** fails, make sure of provisioning the device manually.

If you initiate ZTP, using the **ztp initate** command, the ZTP workflow bypasses the exit checks. ZTP will continue running in the background indefinitely. ZTP will exit only if the workflow has completed with a *SUCCESS* or if it is stopped manually. Ensure to stop ZTP using the **ztp terminate** command after initiating it, if ZTP is active/running on the device.