Zero-Touch Provisioning (ZTP) Guide

Version 2019.1.0, 25 November 2019

Registered Address	Support	Sales
26, Kingston Terrace, Princeton, New Jersey 08540, United States		
		+91 80 4850 5445
http://www.rtbrick.com	support@rtbrick.com	sales@rtbrick.com

Table of Contents

1. Introduction to ZTP

A major goal in any network is a high level of automation. This includes the automatic provisioning of switches newly installed in the network, a process known as *Zero-Touch Provisioning (ZTP)*.

A new switch comes preinstalled with the Open Network Installation Environment (ONIE). The ONIE is an open source *installation environment* that acts as an enhanced boot loader utilizing facilities in a Linux/BusyBox environment. This small Linux operating system allows end-users and channel partners to install the target network OS as part of provisioning.

Because ONIE needs the ability to obtain configuration and image binaries through the management interface, a management LAN is required.

ONIE has access only to the management interface. ONIE starts a DHCP-based discovery process to obtain basic configuration information, such as the management IP address and the URL of the image to install on the switch. ONIE pulls the image and boots it, all without manual intervention.

However, even after the ONIE boots the image, the switch is not yet configured. For configuration, the RtBrick images come with some preinstalled daemons. One, the preinstalled Control Daemon (CtrlD), is responsible for the management of the switch, and another preinstalled daemon, the ZTP Daemon (ZTPD), is the daemon which takes over after the image is activated. The ZTP daemon is responsible for configuring the switch properly.

To do this, hardware box needs to connect to a DHCP server and a management server through the management LAN. The management server is responsible for providing the image binaries and the configuration of each device.

In summary, there are three basic steps to ZTP operation:

1. Discover the IP address through DHCP.
2. Get the image with HTTP.
3. Activate the image configuration and boot it, also with HTTP.

2. ZTP in a Nutshell

This document provides a brief overview of Zero-Touch Provisioning (ZTP) and discusses the managed ZTP process as a ZTP application in the RtBrick Management System (RBMS). The focus is on the REST APIs of the RBMS ZTP application itself, and the management server hosting the actual ZTP service.

Some of these sections discuss how to provision the management server. By the end of this document, we show how ONIE fetches the images.

2.1. The ZTP Process

ZTP is a DHCP-based process driven by ONIE. ONIE is preinstalled on a white box switch. Figure 1 shows the overall ZTP process.

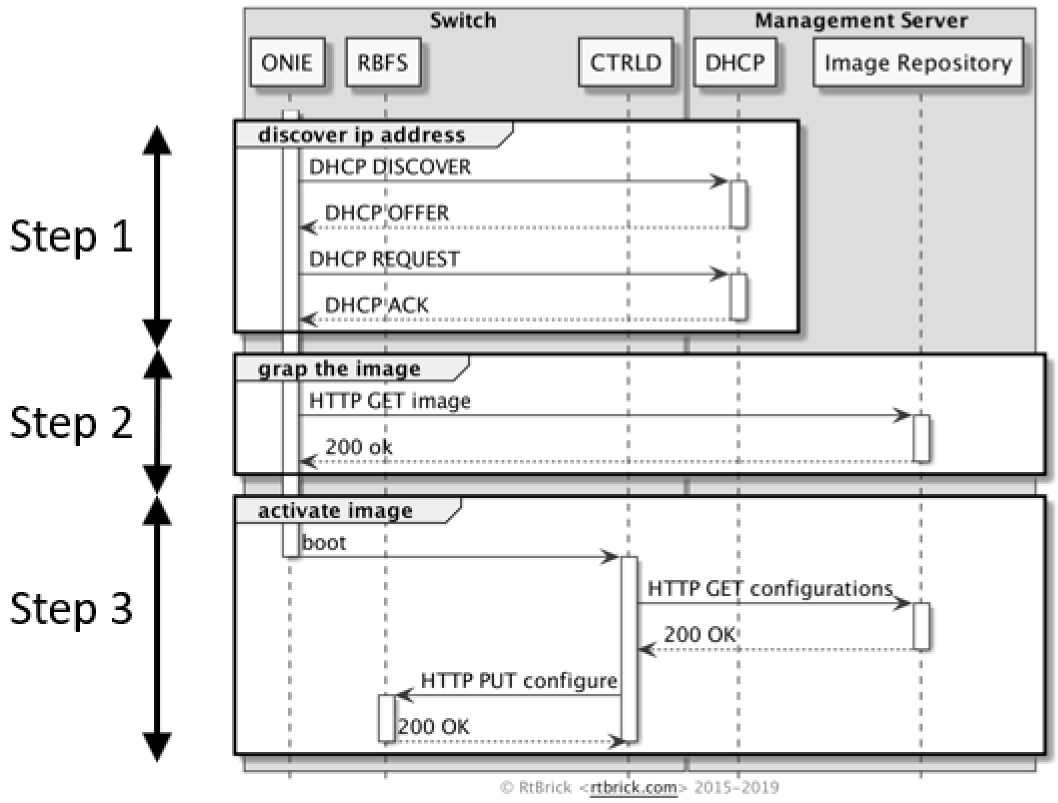

Figure 1. The Overall ZTP Process

The management server runs a DHCP service and a local image repository accessible through HTTP. The complete ZTP process takes place in three distinct steps.

1. ONIE uses DHCP to discover the IP address along with basic device configuration.

2. ONIE determines the image download URL based on the provided DHCP options. For download, ONIE allows different ways to pull an image from the repository. In this ZTP process, HTTP is used to pull the image because ONIE conveys the serial number as the HTTP header. This serial number allows the image repository to identify the switch and select the appropriate image
3. The final step is to activate the image and the RBFS container. The container in turn fetches the application configuration from the image repository.

3. The ZTP Management Server

The management server is responsible for bootstrapping a hardware device, such as a white-box switch.

3.1. Management Server Architecture

Two services are provided by the management server, as required by ONIE:

- The DHCP service assigns an IP address to the switch.
- The HTTP service delivers the NOS image and startup configuration.

This architecture leverages standard software to implement both services, and to configure them to satisfy the ONIE requirements. Figure 2 illustrates the management server components as well as the interactions with RBFS, RBMS, and image repository.

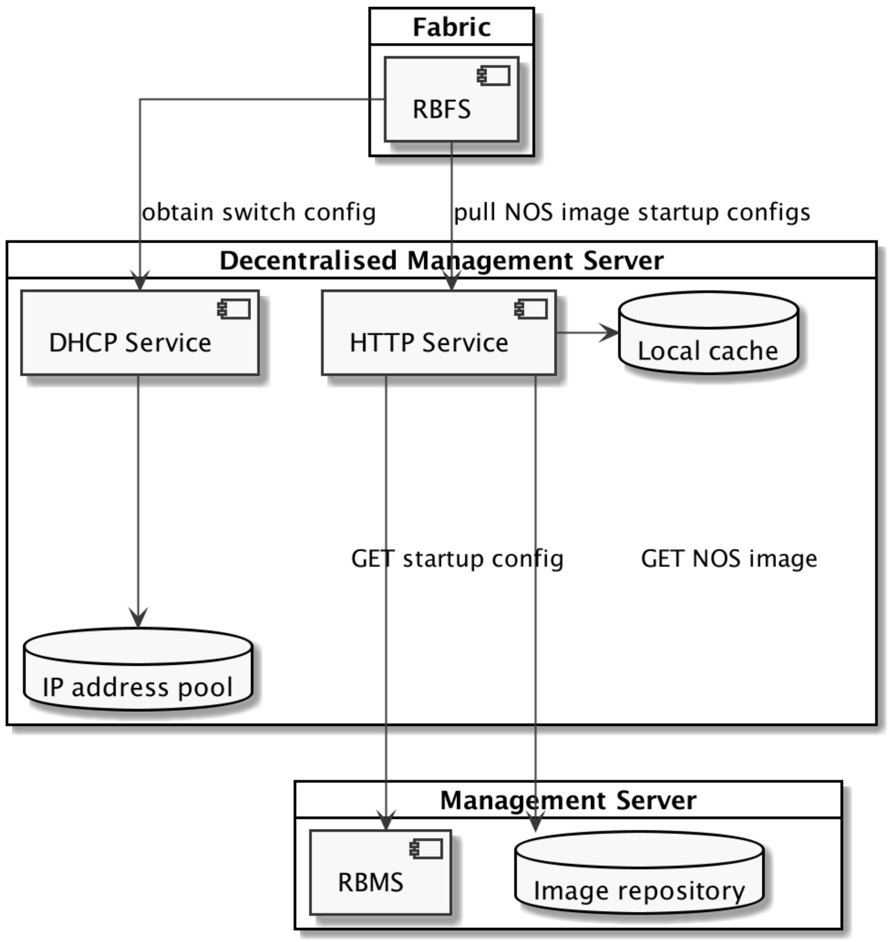

Figure 2. The ZTP Management Server Architecture

3.2. DHCP Service for ZTP

Because of its low set of requirements, the default DHCP server shipped with ubuntu, isc-dhcp, is used to run the DHCP service.

3.3. HTTP Service for ZTP

The HTTP service is implemented as a *transparent proxy*. The proxy tries to pull images and configuration from the image repository and the RBMS respectively. If the image repository or RBMS are not reachable, then the proxy uses a local fallback.

The HTTP service is implemented using *nginx*, an open-source HTTP Server. The nginx is configured to read the **ONIE_SERIAL_NUMBER HTTP** header and maps the serial number to the NOS installer image download path. The mapping configuration is generated by the management server provisioning scripts.

Nginx supports modularized configuration. This means that the mapping configuration can be generated and isolated from the general nginx configuration, but included when needed. This helps configuration management because only the mapping configuration needs to be generated.

4. The ZTP Daemon (ZTPD)

The ZTPD is a *post-ZTP daemon*, which means that it runs after the image is activated. This daemon is responsible for configuring the switch properly.

4.1. Running ZTPD

In a production environment, the ztpd binary starts with default parameters. The service is called rtbrick-ztpd. Key parameters for ZTPD are shown below.

```
$ ztpd -h
Usage of ztpd:
  -addr string
        HTTP network address (default "localhost:19092")
  -dhcp
        Runs dhcp logic
  -onl string
        onl base folder (default "/lib/platform-config/current/onl")
  -serial
        Runs serial logic
  -syslog string
        syslog file (default "/var/log/syslog")
  -run
      Runs the whole ztp logic
  -e string
      Element Name only needed if run is active
  -c string
      Container Name, only needed if run is active
  -version
        Returns the software version
```

To get the installed version, use the ztpd -version command.

4.2. ZTPD-CTRLD-RBFS Inter-Process Communication

Figure 3 shows the inter-process communication between ZTPD and the service that ZTPD relies on., CTRLD and RBFS.

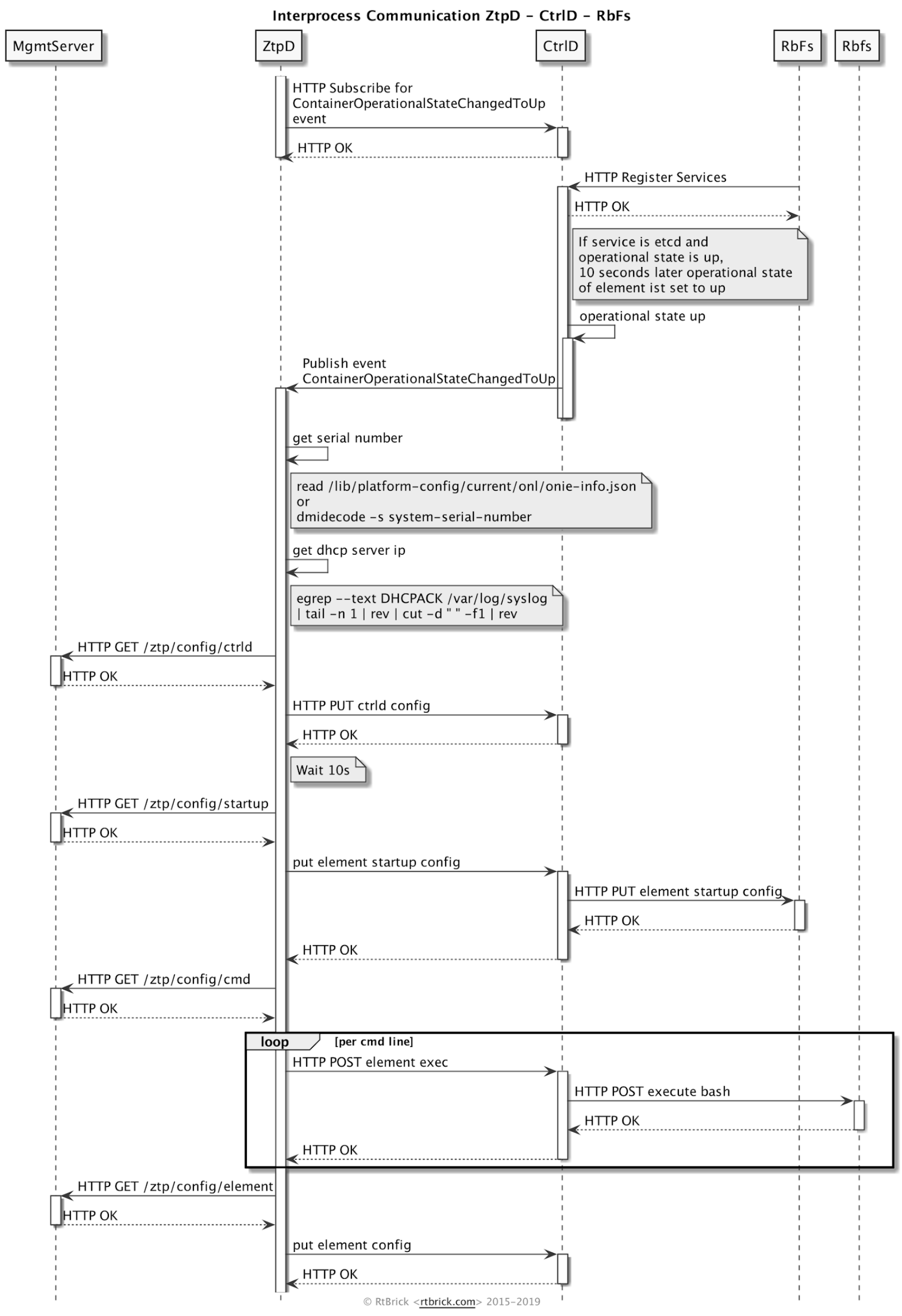
Interprocess Communication ZtpD - CtrlD - RbFs
MgmtServer
ZtpD
CtrlD
RbFs
Rbfs
HTTP Subscribe for ContainerOperationalStateChangedToUp event
HTTP OK
HTTP Register Services
HTTP OK
If service is etcd and operational state is up, 10 seconds later operational state of element ist set to up
operational state up
Publish event ContainerOperationalStateChangedToUp
get serial number
read /lib/platform-config/current/onl/onie-info.json or dmidecode -s system-serial-number
get dhcp server ip
egrep --text DHCPACK /var/log/syslog | tail -n 1 | rev | cut -d " " -f1 | rev
HTTP GET /ztp/config/ctrld
HTTP OK
HTTP PUT ctrld config
HTTP OK
Wait 10s
HTTP GET /ztp/config/startup
HTTP OK
put element startup config
HTTP PUT element startup config
HTTP OK
HTTP OK
HTTP GET /ztp/config/cmd
HTTP OK
loop [per cmd line]
HTTP POST element exec
HTTP POST execute bash
HTTP OK
HTTP OK
HTTP GET /ztp/config/element
HTTP OK
put element config
HTTP OK
© RtBrick <rtbrick.com> 2015-2019

Figure 3. ZTPD Inter-Process Communication with CTRLD and RBFS

ZTPD subscribes to CTRLD for the event “ContainerOperationalStateChangedToUp.” This means that ZTPD is informed when an RtBrick container has an operational state of up.

When the container is up, ZTPD tries to configure the container.

The management server stores the configurations and provides the IP address of the DHCP server. Because all configurations are bound to the serial number, the device serial number is also needed to get the right configuration for the box.

The configurations are downloaded from the management server and the configuring is performed by CtrlD.

It is important to note that the subscription from ZTP to CtrlD is periodic. That is, communication takes place every 60 seconds. This repetition is needed because the subscription is stateless, meaning CtrlD stores the subscription information only in memory. After a restart of CtrlD this information is gone, and a new subscription must be done.

This is a well-known pattern in the microservices domain, and helps promote better resilience of the ecosystem.

The following configurations are stored at the management server:

1. Startup Configuration
2. Cmd Configuration
3. Element Configuration
4. CTRLD Configuration

4.2.1. Startup Configuration

The startup configuration is the running-configuration of the RBFS switch.

This configuration must be reachable at <dhcpip>/ztp/config/startup

4.2.2. Cmd Configuration

ZTPD can load a text or plain file from the <dhcpip>/ztp/config/ctrld location.

This file contains commands that are executed in the RBFS bash.

Here is an example file:

```
ls -la | grep var
rtb confd show running-configuration
```

Each line is executed against the RBFS.

4.2.3. Element Configuration

The element config must be reachable at <dhcpip>/ztp/config/element.

This configuration sets the element name and the PoD name in the container. The element name is used to identify the container to the RBMS.

Here is an example file:

```
{
    "element_name": "element_name",
    "pod_name": "pod2"
}
```

4.2.4. CTRLD Configuration

Th e CtrlD must be reachable at <dhcpip>/ztp/config/ctrld.

A detailed description of this configuration can be found in the CtrlD documentation.

4.3. Log Files

The relevant log files can be found here:

- /var/log/rtbrick-ztpd.log