cloudboot community edition documentation en Documentation

Release 0.0.1

iDCOS

Sep 19, 2017

Table of Contents:

CHAPTER 1

Preface

Background

Bare-metal provisioning is one of most import activities of IT operation, and usually it is done by hand, using OS DVD or USB. Given the variety of server vendors, as well as the complexity of hardware and S/W configuration, bare-metal provisioning is considerred as a time-consuming, error-prone IT process performed only by high skill IT professionals. To meet the ever-higher demands for physical servers in enterprise, it is cirtical to automate OS provisioning in large scale.

Purpose

Different server vendors have diffent ways to configure the physical server, and not compatible with each other. To automate the server hardware configureation and OS provisioning process, we propose unified standard and build a tool upon it, we call it *Cloudboot*.

This document describes the architecture, usage and API of Cloudboot.

Target Audience

- IT professionals
- IT automation software developer and integrators
- x86 server vendors

Contents of this document

Chapter	Content
Chapter One	Archicture, modules and upgrade process
Chapter Two	Requirements and installation
Chapter Three	BootOS usage, development and upgrade
Chapter Four	hardware configuration specification and development guide
Chapter Five	OS template
Chapter Six	Virtual machine management
Chapter Seven	Cloudboot API
Appendix	Other useful informations

Github Repo

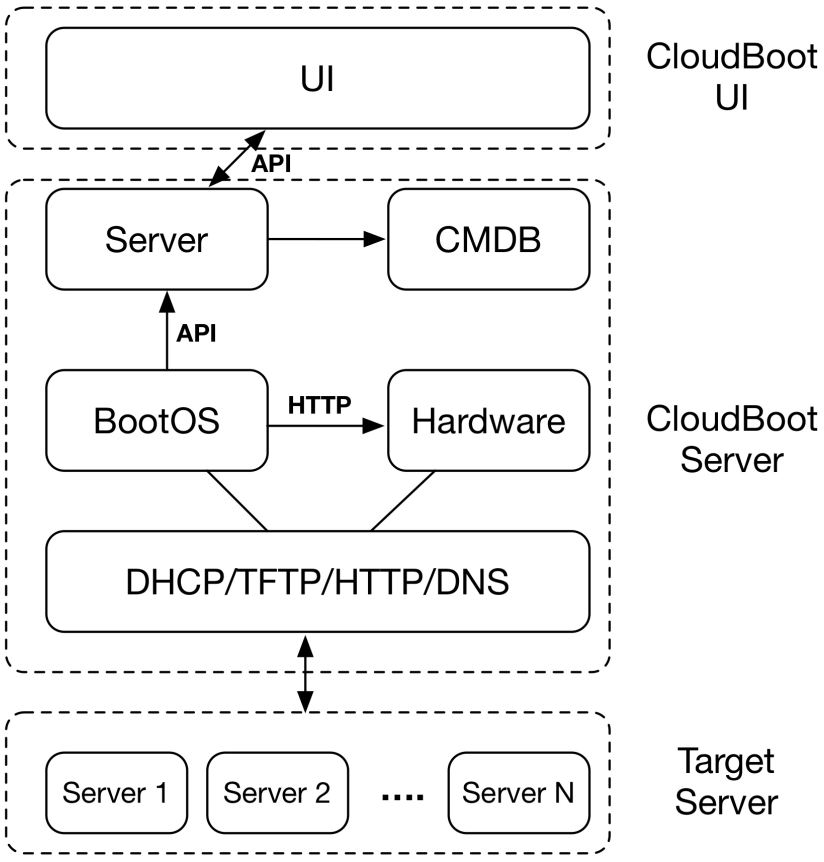
Cloudboot is opensouce, you may find Cloudboot source code at:

https://github.com/idcos/osinstall

CHAPTER 2

Cloudboot Architecture

Architecture Diagram



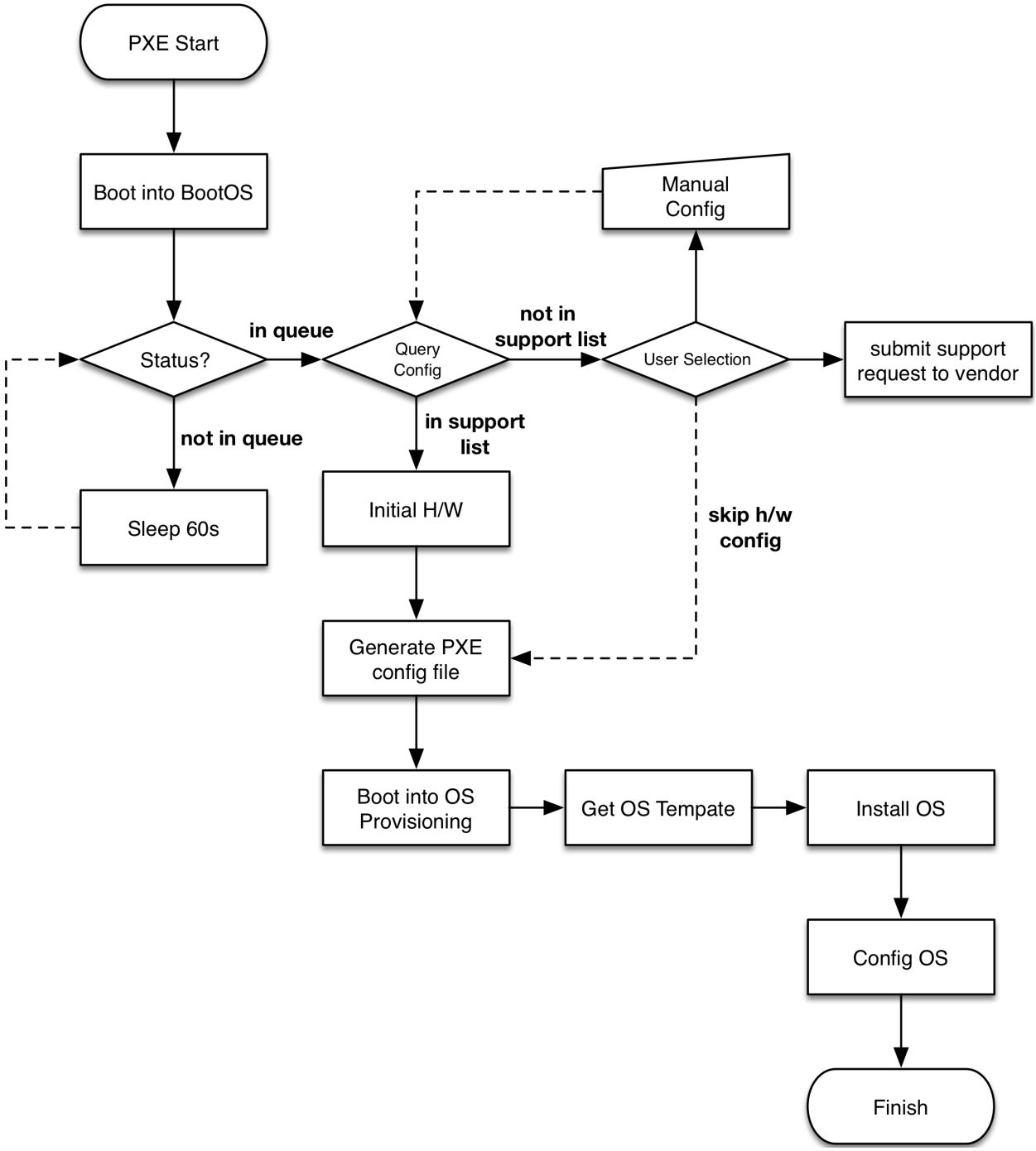
As shown above, Cloudboot has six functional modules. Modules are oosely-coupled and communicate with each other via API.

Functional Modules

Module Name	Description
UI	web-based user interface, for user input, information display and query such as provision progress, operation logs, search, etc.
Server	provides API service to UI module
Hardware	hardware configuration and management module
CMDB	lightweight configuration management system for server hardware, OS templates, configuration and user information
BootOS	boot os with config agent to collect and config hardware settings of target server
Base Service	provide base service like dhcp/tftp/http/dns/samba for OS provision

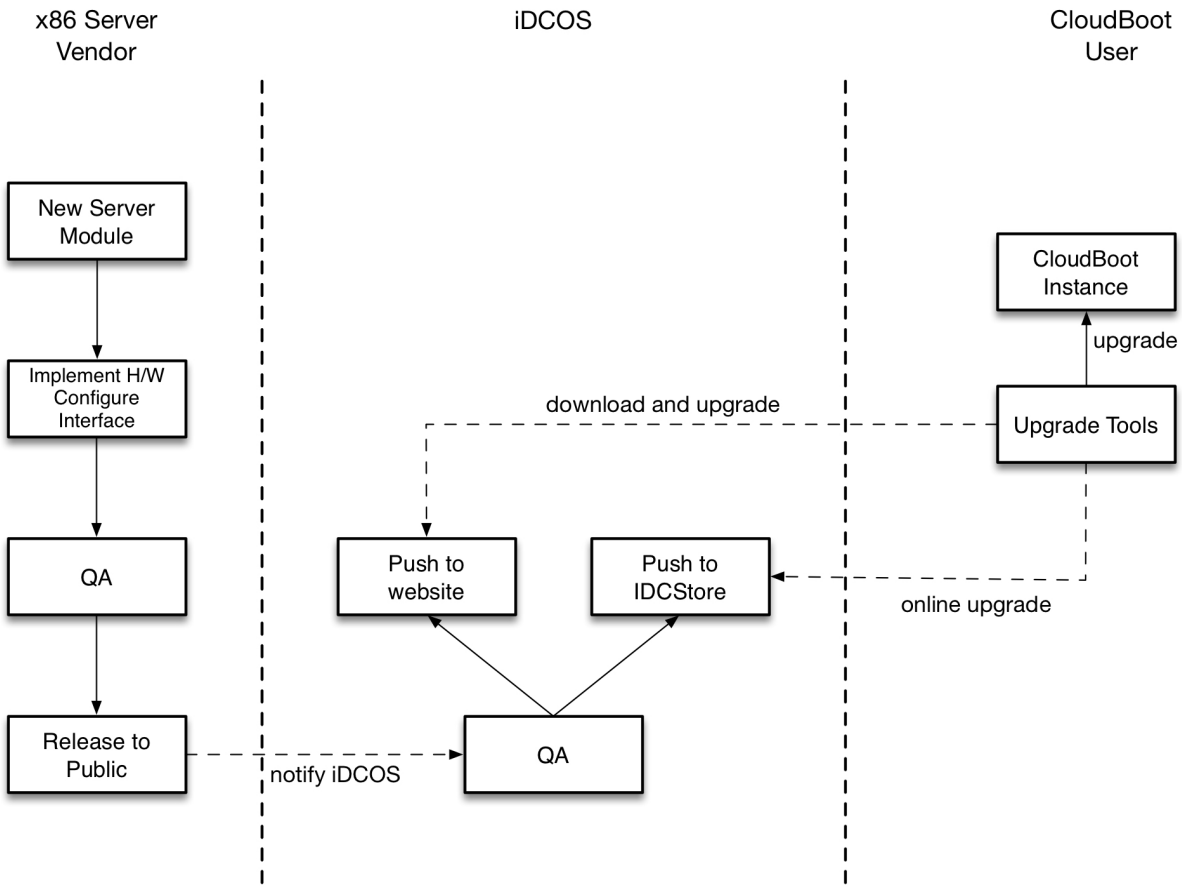
OS Provision Workflow

1. user submit OS provisioning task via web UI
2. boot server manually or by remote control interface
3. target server boot into BootOS
4. agent in BootOS start automatically, check if target server is in provision queue
5. if not, sleep 60 seconds, then check again until server is listed in provision queue
6. *Server* check if target supports hardware configuration
7. if not, notify user to configure hardware manually
8. *BootOS agent* fetch configuration template from *Server* and perform hardware configuration
9. *BootOS agent* upload MAC address information, *Server* generate pxe related file
10. *BootOS agent* boot target into network boot mode, start OS provisioning
11. Install OS according to OS templage
12. Run post installation script
13. Update provisioning progress, update CMDB



New Hardware Support Workflow

Cloudboot configures server hardware via open standard interface, any x86 server compatible with the interface can be configured automatically. To support hardware configuration of new server module: 1. x86 server vendor submit hardware configuration implementation to CloudJ/iDCOS 2. iDCOS verify the implementation and push the update to IDCStore 3. End user update Cloudboot with new supported server online, or install it by offline package


x86 Server Vendor
iDCOS
CloudBoot User
New Server Module
Implement H/W Configure Interface
QA
Release to Public
notify iDCOS
QA
Push to website
Push to IDCStore
download and upgrade
online upgrade
Upgrade Tools
upgrade
CloudBoot Instance

CHAPTER 3

Installation

System Requirements

Cloudboot requires RHEL/Centos 6.0 or higher.

While the system can operate with lower specs, these are the recommendations for the best experience while testing or deploying Cloudboot:

Testing	Production
• Dual CPU system • 2GB RAM • 100GB storage	• Quad core CPU system • 8GB RAM • 200GB storage

Note: the storage requirement includes disk space needed for OS media (OS distribution). In real world deploayment, OS media is normally mounted from external/network locations.

System Configuration

- Turn off OS firewall:

```
service iptables stop
chkconfig iptables off
```

- Turn off selinux (requires OS reboot):

```
sed -i.bak '/^SELINUX=/cSELINUX=disabled' /etc/sysconfig/selinux
```

Install Cloudboot via RPM Package

- Download Cloudboot RPM from http://www.idcos.com/store/x86
- Install RPM Package:

```
rpm -ivh cloudboot-$version.x86_64.rpm
Preparing...                ########################################### [100%]
1:cloudboot              ########################################### [100%]
```

Setup Cloudboot

Add DCHP subnet

```
# cat /opt/cloudboot/etc/dhcp/dhcpd.conf
allow booting;
allow bootp;
ddns-update-style none;
ping-check true;
ping-timeout 3;
default-lease-time 120;
max-lease-time 600;
authoritative;
next-server osinstall.idcos.com;
filename "undionly.kkpxe";
option domain-name "idcos.com";
option domain-name-servers 192.168.0.1;
option root-path "osinstall.idcos.com:/";

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.101 192.168.0.200;
    option routers 192.168.0.1;
}
```

Note: Please adjust DHCP subnet setting according to target environment.

Import OS Media

Take Centos 6.7 for example:

```
mount -o loop CentOS-6.7-x86_64-bin-DVD1.iso /media
rsync -az /media/ /opt/cloudboot/home/www/centos/6.7/os/x86_64/
umount /media
```

Start Cloudboot Service

```
service cloudboot start
Starting dhcpd:                                            [  OK  ]
Starting dnsmasq:                                          [  OK  ]
Starting xinetd:                                           [  OK  ]
Starting nginx:                                            [  OK  ]
Starting SMB services:                                     [  OK  ]
Starting mysqld:                                           [  OK  ]
Starting cloudboot-server:                                 [  OK  ]
```

CHAPTER 4

BootOS

About BootOS

BootOS is a in-memory operation system. After loaded into system, BootOS could perform hardware configuration tasks such as:

- upgrade server BIOS
- modify RAID setting
- configure OOB
- disk partition
- create file system

How It Works

server power on -> start PXE process -> load BootOS kernel -> load initrd -> extract rootfs -> switch to rootfs -> run init -> start agent -> start shell

Functions

- build-in agent to perform server configuration and provisioning tasks
- collect hardware information and upload to Cloudboot server

- receive configuration template from Cloudboot server and perform hardware configuration task on target server
- upload MAC address information to Cloudboot server for PXE setting auto generation

Features

- customized in-memory OS based on centos 6.7 kernel, includes latest hardware support
- include rpm/yum for easy extend
- running entirely in memory, donot store data on disk
- be able to collection and configure server hardware information such as RAIN/OOB/BIOS, etc.

Configure BootOS

Basic Configure

Configure boot environment for BootOS:

```
# cat /opt/cloudboot/var/lib/tftpboot/pxelinux.cfg/default
DEFAULT menu.c32
PROMPT 0
TIMEOUT 30

LABEL bootos
MENU LABEL ^BootOS
MENU DEFAULT
KERNEL http://osinstall.idcos.com/bootos/vmlinuz
APPEND initrd=http://osinstall.idcos.com/bootos/initrd.img console=tty0 selinux=0␣
↪biosdevname=0 SERVER_ADDR=http://osinstall.idcos.com
IPAPPEND 2
```

Parameter Note:

- `TIMEOUT 30`, wait for 3 seconds before boot into BootOS
- load vmlinuz and initrd.img by http instead of tftp
- `biosdevname=0`, force network adapter naming to follow ethX standard
- `SERVER_ADDR=http://osinstall.idcos.com`, callback URL, adjust according to target environment
- `IPAPPEND 2`, force naming PXE network adapter as eth0

Note: **SERVER_ADDR** and **IPAPPEND 2** are required for BootOS boot properly.

Developer Mode

The difference between user mode and developer mode is:

- in user mode, after BootOS agent collect and upload hardware informatin to Cloudboot server, Cloudboot server will check if it in Cloudboot's support list, if not, BootOS agent will *NOT* perform hardware configuration on target server.
- in developer mode, BootOS agent will *ALWAYS* perform hardware configuration on target server.

To switch to developer mode, add `DEVELOPER=1` setting in PXE configuration file, as following:

```
# cat /opt/cloudboot/var/lib/tftpboot/pxelinux.cfg/default
DEFAULT menu.c32
PROMPT 0
TIMEOUT 30

LABEL bootos
MENU LABEL ^BootOS
MENU DEFAULT
KERNEL http://osinstall.idcos.com/bootos/vmlinuz
APPEND initrd=http://osinstall.idcos.com/bootos/initrd.img console=tty0 selinux=0␣
↪biosdevname=0 SERVER_ADDR=http://osinstall.idcos.com DEVELOPER=1
IPAPPEND 2
```

Advance Options

Use following options to customize BootOS agent behavior:

```
# cat /opt/cloudboot/var/lib/tftpboot/pxelinux.cfg/default
DEFAULT menu.c32
PROMPT 0
TIMEOUT 30

LABEL bootos
MENU LABEL ^BootOS
MENU DEFAULT
KERNEL http://osinstall.idcos.com/bootos/vmlinuz
APPEND initrd=http://osinstall.idcos.com/bootos/initrd.img console=tty0 selinux=0␣
↪biosdevname=0 SERVER_ADDR=http://osinstall.idcos.com PRE=http://osinstall.idcos.com/
↪pre.sh POST=http://osinstall.idcos.com/post.py
IPAPPEND 2
```

Note

- `PRE=http://osinstall.idcos.com/pre.sh`, run *pre.sh* right after agent started
- `POST=http://osinstall.idcos.com/post.py`, run *post.py* before system reboot

Note: pre/post action supports script file as well as binary file

---

Driver Update

Note: User can update BootOS by apply CloudBoot updates, below update process is about update BootOS driver manually

Browse Existing Drivers

BootOS is based on Centos and the driver folder remains the same, take network driver for example:

```
# ls /lib/modules/`uname -r`/kernel/drivers/net/
3c59x.ko      b44.ko       cnic.ko    e100.ko        ifb.ko       macvtap.ko      netxen
↪       ppp_generic.ko  r6040.ko    slhc.ko         tg3.ko           virtio_net.ko
8139cp.ko     benet        cxgb3      enic           igb          mdio.ko         niu.ko
↪       ppp_mppe.ko     r8169.ko    slip.ko         tlan.ko          vmxnet3
8139too.ko    bna          cxgb4      epic100.ko     igbvf        mii.ko          ns83820.
↪ko       pppoe.ko        s2io.ko     smsc9420.ko    tulip            vxge
8390.ko       bnx2.ko      cxgb4vf    ethoc.ko       ipg.ko       mlx4            pch_gbe
↪       pppol2tp.ko     sc92031.ko  starfire.ko    tun.ko           vxlan.ko
acenic.ko     bnx2x        dl2k.ko    fealnx.ko      ixgb         mlx5            pcmcia
↪       pppox.ko        sfc          sundance.ko    typhoon.ko       wan
amd8111e.ko   bonding      dnet.ko    forcedeth.ko   ixgbe        myri10ge        pcnet32.
↪ko       ppp_synctty.ko  sis190.ko   sungem.ko      usb              wimax
atl1c         can          dummy.ko   hyperv         ixgbevf      natsemi.ko      phy
↪       qla3xxx.ko      sis900.ko   sungem_phy.ko  veth.ko          wireless
atl1e         cassini.ko   e1000      i40e           jme.ko       ne2k-pci.ko     ppp_async.
↪ko     qlcnic          skge.ko     sunhme.ko      via-rhine.ko     xen-netfront.ko
atlx          chelsio      e1000e     i40evf         macvlan.ko   netconsole.ko   ppp_
↪deflate.ko   qlge             sky2.ko     tehuti.ko      via-velocity.ko
```

Update Driver

Take igb driver for example, first, you need to install developer tools, kernel headers and related libraries:

```
yum install kernel-devel kernel-headers gcc make rpm-build wget
```

Then download latest igb dirver source code and compile:

```
mkdir -p /root/rpmbuild/SOURCES/
wget -P /root/rpmbuild/SOURCES/ https://downloadmirror.intel.com/13663/eng/igb-5.3.2.
↪tar.gz
tar zxpf /root/rpmbuild/SOURCES/igb-5.3.2.tar.gz
rpmbuild -bb igb-5.3.2/igb.spec
```

Finally copy generated rpm file `/root/rpmbuild/RPMS/x86_64/igb-5.3.2-1.x86_64.rpm` into BootOS.

CHAPTER 5

Hardware Configuration

Cloudboot supports automate hardware configuration as part of provisioning process. To support automate hardware configuration, x86 server vendors needs to provide a set of vendor scripts, which will be executed by BootOS agent during the provisioning.

Hardware Configuration Basics

Server Hardware Settings

Server hardware settings includes RAID, OOB and BIOS:

- RAID: RAID 0 / RAID 1 / RAID 5 / RAID 10
- OOB: user/password/network/license
- BIOS: VT / HT / NUMA / C-State / Turbo, etc settings

Vendor Tools

Different server vendors have its own tools to configure server hardware, for instance:

Vendor	Tools	Description
Dell	MegaCli	LSI MegaRAID configuration
Dell	syscfg	configure BIOS
Dell	racadm	configure DRAC
Dell	dsu	configure firmware
HP	hpacucli	configure HP Smart Array Raid
HP	conrep	configure BIOS
HP	hponcfg	configure iLO
HP	firmware-tools	Online ROM Flash upgrade
Lenovo	MegaRAID	configure RAID
Lenovo	asu	configure BIOS

Standardized Toolset

About

Cloudboot provides a standardized hardware configuration toolset, which includes configuration scripts for mainstream x86 server vendors. Toolset is free and opensource, you may use rsync to download it to your Cloudboot instance:

```
rsync -azHP --delete mirror.idcos.com::hw-tools /opt/cloudboot/home/www/hw-tools
```

The toolset source code can be found at https://github.com/idcos/osinstall-hardware

RAID Configuration Script

Cloudboot standardlize RAID configuration interface by wrap it into `raid.sh` script:

```
# /opt/yunji/osinstall/dell/raid.sh
raid.sh: raid config tool
Usage: raid.sh [OPTION...]
-c, --clear                          Clear raid config
-r, --raid 0/1/5/10                  Raid level for disk
-d, --disk [0|1,2|3-5|6-|all]        Disk slot num
-H, --hotspare [0|1,2|3-5|6-|all]    Hotspare disk slot num
-i, --init                           Initialize all disk
-h, --help                           Show this help message
```

Parameters:

- -c clean old configuration
- -r set RAID level, support RAID 0/1/5/10
- -d disk ID, supports single disk, multiple disks (seperated by comma), disk ID range (for instance, `3-5` means disk ID 3,4,5, `6-` means all disks with id equal or greater than 6)

`raid.sh` usage examples:

- create RAID 0 on all disks, then init disks:

```
/opt/yunji/osinstall/dell/raid.sh -c -r 10 -d all -i
```

- create RAID 1 using first two disks, then create RAID 5 using rest of disks:

```
/opt/yunji/osinstall/dell/raid.sh -c -r 1 -d 0,1
/opt/yunji/osinstall/dell/raid.sh -r 5 -d 2- -i
```

- create RAID 5 using first 4 disks, then create HotSpare using disk 5:

```
/opt/yunji/osinstall/dell/raid.sh -c -r 5 -d 0-4
/opt/yunji/osinstall/dell/raid.sh -d 5 -i
```

Create Configure Script

All vendor specific hardware configure script is located under /opt/yunji/osinstall/**vendor/** folder. Take Dell script for example:

RAID script

```
# /opt/yunji/osinstall/dell/raid.sh
raid.sh: raid config tool
Usage: raid.sh [OPTION...]
-c, --clear         Clear raid config
-l, --level         Raid level for all disk
-s, --size          Set size (default MB) of virtual drive

Help options:
-h, --help          Show this help message
```

OOB Script

```
# /opt/yunji/osinstall/dell/oob.sh
oob.sh: oob config tool
Usage: oob.sh [OPTION...]
-n, --network       Set the IP address source
-i, --ip            Set the IP address
-m, --netmask       Set the Subnet Mask
-g, --gateway       Set the Default Gateway IP
-u, --username      Enable user access mode for userid
-p, --password      Set the user password
-r, --reset         Instructs the BMC to perform a cold reset

Help options:
-h, --help          Show this help message
```

BIOS Script

```
# /opt/yunji/osinstall/dell/bios.sh
bios.sh: dell bios config tool
Usage: bios.sh [OPTION...]
-t, --virtualization          Enable or disabled Virtualization Technology
-c, --cstates                 Enable or disabled CPU C-States
```

```
Help options:
-h, --help          Show this help message
```

Packaging Standard

RPM Dependence

Specify RPM dependence in SPEC file, take Dell script for example:

- RAID: depends on `MegaCli`
- OOB: depends on `ipmitool`
- BIOS: depends on `syscfg`

Script Path

Locate script in vendor folder, for instance:

```
# tree
.
|-- dell
|   |-- bios.sh
|   |-- oob.sh
|   `-- raid.sh
|-- hp
|   |-- bios.sh
|   |-- oob.sh
|   `-- raid.sh
`-- inspur
    |-- bios.sh
    |-- oob.sh
    `-- raid.sh
```

Query script path in RPM:

```
rpm -ql dell-hw-tools
```

Command output:

```
/opt/yunji/osinstall/dell/bios.sh
/opt/yunji/osinstall/dell/oob.sh
/opt/yunji/osinstall/dell/raid.sh
```

SPEC File

Create SPEC file for RPM generation, for instance:

```
# cat dell-hw-tools.spec
%define __spec_prep_post true
%define __spec_prep_pre true
%define __spec_build_post true
%define __spec_build_pre true
%define __spec_install_post true
%define __spec_install_pre true
%define __spec_clean_post true
%define __spec_clean_pre true
%define _binary_filedigest_algorithm 1
%define _build_binary_file_digest_algo 1
%define _binary_payload w9.gzdio

Name: dell-hw-tools
Version: 0.1
Release: 1
Summary: none
AutoReqProv: no
BuildRoot: %buildroot
Prefix: /opt/yunji/osinstall/dell
Group: default
License: GPLv3+
Vendor: CentOS
URL: none
Packager: admin@dell.com

Requires: MegaCli
Requires: ipmitool
Requires: syscfg

%Note
none

%prep

%build

%install

%clean

%files
%defattr(-,root,root,-)
/opt/yunji/osinstall/dell/oob.sh
/opt/yunji/osinstall/dell/raid.sh
/opt/yunji/osinstall/dell/bios.sh

%changelog
```

Create RPM

Build RPM by `rpmbuild` command with SPEC configuration file:

```
rpmbuild -bb dell-hw-tools.spec
```

Command output:

```
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.SyB7Tz
Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.NR8Yga
Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.O7eNEK
Processing files: dell-hw-tools-0.1-1.x86_64
Checking for unpackaged file(s): /usr/lib/rpm/check-files /root/rpmbuild/BUILDROOT/
↪dell-hw-tools-0.1-1.x86_64
Wrote: /root/rpmbuild/RPMS/x86_64/dell-hw-tools-0.1-1.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.OoeaBV
```

Test RPM

To test RPM package generated by `rpmbuild`, run the following command in BootOS:

```
yum install dell-hw-tools
```

Command output:

```
Loaded plugins: fastestmirror, security
Setting up Install Process
Loading mirror speeds from cached hostfile
base
↪                                                                | 3.7 kB     ⌊
↪00:00
extras
↪                                                                | 3.4 kB     ⌊
↪00:00
idcos
↪                                                                | 2.9 kB     ⌊
↪00:00
idcos/primary_db
↪                                                                |  40 kB     ⌊
↪00:00
updates
↪                                                                | 3.4 kB     ⌊
↪00:00
updates/primary_db
↪                                                                | 2.7 MB     ⌊
↪00:02
Resolving Dependencies
--> Running transaction check
---> Package dell-hw-tools.x86_64 0:0.1-1 will be installed
--> Processing Dependency: syscfg for package: dell-hw-tools-0.2-1.x86_64
--> Processing Dependency: ipmitool for package: dell-hw-tools-0.2-1.x86_64
--> Processing Dependency: MegaCli for package: dell-hw-tools-0.2-1.x86_64
--> Running transaction check
---> Package MegaCli.noarch 0:8.07.10-1 will be installed
---> Package ipmitool.x86_64 0:1.8.11-29.el6_7 will be installed
---> Package syscfg.x86_64 0:5.1.0-4.70.1.el6 will be installed
--> Processing Dependency: srvadmin-isvc for package: syscfg-5.1.0-4.70.1.el6.x86_64
--> Processing Dependency: srvadmin-hapi for package: syscfg-5.1.0-4.70.1.el6.x86_64
--> Processing Dependency: srvadmin-deng for package: syscfg-5.1.0-4.70.1.el6.x86_64
--> Processing Dependency: libdchipm.so.8()(64bit) for package: syscfg-5.1.0-4.70.1.
↪el6.x86_64
--> Processing Dependency: libdchbas.so.8()(64bit) for package: syscfg-5.1.0-4.70.1.
↪el6.x86_64
--> Running transaction check
---> Package srvadmin-deng.x86_64 0:8.1.0-4.8.1.el6 will be installed
```

```
--> Processing Dependency: srvadmin-omilcore for package: srvadmin-deng-8.1.0-4.8.1.
↪el6.x86_64
--> Processing Dependency: srvadmin-omilcore for package: srvadmin-deng-8.1.0-4.8.1.
↪el6.x86_64
---> Package srvadmin-hapi.x86_64 0:8.1.0-4.10.2.el6 will be installed
---> Package srvadmin-isvc.x86_64 0:8.1.0-4.38.1.el6 will be installed
--> Running transaction check
---> Package srvadmin-omilcore.x86_64 0:8.1.0-4.85.1.el6 will be installed
--> Processing Dependency: smbios-utils-bin for package: srvadmin-omilcore-8.1.0-4.85.
↪1.el6.x86_64
--> Running transaction check
---> Package smbios-utils-bin.x86_64 0:2.2.27-4.4.1.el6 will be installed
--> Processing Dependency: libsmbios = 2.2.27-4.4.1.el6 for package: smbios-utils-bin-
↪2.2.27-4.4.1.el6.x86_64
--> Processing Dependency: libsmbios_c.so.2()(64bit) for package: smbios-utils-bin-2.
↪2.27-4.4.1.el6.x86_64
--> Processing Dependency: libsmbios.so.2()(64bit) for package: smbios-utils-bin-2.2.
↪27-4.4.1.el6.x86_64
--> Running transaction check
---> Package libsmbios.x86_64 0:2.2.27-4.4.1.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
Package                                 Arch
↪Version                                    Repository                      ⌊
↪    Size
================================================================================
Installing:
dell-hw-tools                           x86_64                          0.
↪1-1                                      idcos                           ⌊
↪3.3 k
Installing for dependencies:
MegaCli                                 noarch                          8.
↪07.10-1                                  idcos                           ⌊
↪1.5 M
ipmitool                                x86_64                          1.
↪8.11-29.el6_7                            updates                         ⌊
↪464 k
libsmbios                               x86_64                          2.
↪2.27-4.4.1.el6                           idcos                           ⌊
↪2.0 M
smbios-utils-bin                        x86_64                          2.
↪2.27-4.4.1.el6                           idcos                           ⌊
↪119 k
srvadmin-deng                           x86_64                          8.
↪1.0-4.8.1.el6                            idcos                           ⌊
↪730 k
srvadmin-hapi                           x86_64                          8.
↪1.0-4.10.2.el6                           idcos                           ⌊
↪959 k
srvadmin-isvc                           x86_64                          8.
↪1.0-4.38.1.el6                           idcos                           ⌊
↪7.8 M
srvadmin-omilcore                       x86_64                          8.
↪1.0-4.85.1.el6                           idcos                           ⌊
↪ 29 k
```

```
syscfg                                          x86_64                                  5.
↪1.0-4.70.1.el6                                   idcos                                  ␣
↪436 k

Transaction Summary
================================================================================================
Install      10 Package(s)

Total download size: 14 M
Installed size: 43 M
Is this ok [y/N]:
```

CHAPTER 6

OS Template

About Answer Files

Different OS requires its own answer file to perform provisioning in unattend manner, for instance:

OS	Answer File
RedHat/CentOS	Kickstart
SUSE	AutoYaST
VMware/ESX	Kickstart
Windows	WAIK

BootOS provides a unified web-base UI to custom OS provision answer file along with default settings.

PXE-boot Template

```
DEFAULT centos6.7
LABEL centos6.7
KERNEL http://osinstall.idcos.com/centos/6.7/os/x86_64/images/pxeboot/vmlinuz
APPEND initrd=http://osinstall.idcos.com/centos/6.7/os/x86_64/images/pxeboot/initrd.
 ↪img ksdevice=bootif ks=http://osinstall.idcos.com/api/osinstall/v1/device/
 ↪getSystemBySn?sn={sn} console=tty0 selinux=0 biosdevname=0
IPAPPEND 2
```

Note:

- `ksdevice=bootif` set pxe network adapter, use with `IPAPPEND 2`

Linux Template

Take CentOS 6.7 for example:

```
install
url --url=http://osinstall.idcos.com/centos/6.7/os/x86_64/
lang en_US.UTF-8
keyboard us
network --onboot yes --device bootif --bootproto dhcp --noipv6
rootpw  --iscrypted $6$eAdCfx9hZjVMqyS6
↪$BYIbEu4zeKp0KLnz8rLMdU7sQ5o4hQRv55o151iLX7s2kSq.5RVsteGWJlpPMqIRJ8.
↪WUcbZC3duqX0Rt3unK/
firewall --disabled
authconfig --enableshadow --passalgo=sha512
selinux --disabled
timezone Asia/Shanghai
text
reboot
zerombr
bootloader --location=mbr --append="console=tty0 biosdevname=0 audit=0 selinux=0"
clearpart --all --initlabel
part /boot --fstype=ext4 --size=256 --ondisk=sda
part swap --size=2048 --ondisk=sda
part / --fstype=ext4 --size=100 --grow --ondisk=sda

%packages --ignoremissing
@base
@core
@development

%pre
_sn=$(dmidecode -s system-serial-number 2>/dev/null | awk '/^[^#]/ { print $1 }')
curl -H "Content-Type: application/json" -X POST -d "{\"Sn\":\"$_sn\",\"Title\":\
↪"Start OS Installation\",\"InstallProgress\":0.6,\"InstallLog\":\"SW5zdGFsbCBPUwo=\
↪"}" http://osinstall.idcos.com/api/osinstall/v1/report/deviceInstallInfo
curl -H "Content-Type: application/json" -X POST -d "{\"Sn\":\"$_sn\",\"Title\":\
↪"Disk Partition and Install Software Package\",\"InstallProgress\":0.7,\"InstallLog\
↪":\"SW5zdGFsbCBPUwo=\"}" http://osinstall.idcos.com/api/osinstall/v1/report/
↪deviceInstallInfo

%post
progress() {
    curl -H "Content-Type: application/json" -X POST -d "{\"Sn\":\"$_sn\",\"Title\":\"
↪$1\",\"InstallProgress\":$2,\"InstallLog\":\"$3\"}" http://osinstall.idcos.com/api/
↪osinstall/v1/report/deviceInstallInfo
}

_sn=$(dmidecode -s system-serial-number 2>/dev/null | awk '/^[^#]/ { print $1 }')

progress "hostname and network setting" 0.8 "Y29uZmlnIG5ldHdvcmsK"

# config network
curl -o /tmp/networkinfo "http://osinstall.idcos.com/api/osinstall/v1/device/
↪getNetworkBySn?sn=${_sn}&type=raw"
```

```
source /tmp/networkinfo

cat > /etc/sysconfig/network <<EOF
NETWORKING=yes
HOSTNAME=$HOSTNAME
GATEWAY=$GATEWAY
NOZEROCONF=yes
NETWORKING_IPV6=no
IPV6INIT=no
PEERNTP=no
EOF

cat > /etc/sysconfig/network-scripts/ifcfg-eth0 <<EOF
DEVICE=eth0
BOOTPROTO=static
IPADDR=$IPADDR
NETMASK=$NETMASK
ONBOOT=yes
TYPE=Ethernet
NM_CONTROLLED=no
EOF

progress "Add User" 0.85 "YWRkIHVzZXIgeXVuamkK"
useradd yunji

progress "Configure System Service" 0.9 "Y29uZmlnIHN5c3RlbSBzZXJ2aWNlCg=="

# config service
service=(crond network ntpd rsyslog sshd sysstat)
chkconfig --list | awk '{ print $1 }' | xargs -n1 -I@ chkconfig @ off
echo ${service[@]} | xargs -n1 | xargs -I@ chkconfig @ on

progress "System Settings" 0.95 "Y29uZmlnIGJhc2ggcHJvbXB0Cg=="

# custom bash prompt
cat >> /etc/profile <<'EOF'

export LANG=en_US.UTF8
export PS1='\n\e[1;37m[\e[m\e[1;32m\u\e[m\e[1;33m@\e[m\e[1;
↪35m\H\e[m:\e[4m`pwd`\e[m\e[1;37m]\e[m\e[1;36m\e[m\n\$ '
export HISTTIMEFORMAT='[%F %T] '
EOF

progress "Provision Success!" 1 "aW5zdGFsbCBmaW5pc2hlZAo="
```

Notes:

- `--url=xxx` OS Image URL
- `rootpw --iscrypted` root password setting, generated by `grub-crypt`
- Using `curl` to post progress message to BootOS server,
- After disk partition and software package installation, using `progress` keyword to update progress
- Query network setting from BootOS server via web service API
- Set `progress` to `1` while provision is success

VMWare/ESX Template

- Import ESXi OS Image

```
mount -o loop VMware-VMvisor-Installer-6.0.0.update01-3029758.x86_64.iso /media/
rsync -az /media/ /opt/cloudboot/home/www/esxi/6.0u1/
```

- Modify boot.cfg to use http instead of tftp

```
sed -i.orig -e 's;/;http://osinstall.idcos.com/esxi/6.0u1/;g' -e '/kernelopt/d' /opt/
↪cloudboot/home/www/esxi/6.0u1/boot.cfg
```

Note: Select `esxi6.0u1-x86_64` as pxe template, `esxi6.0` as OS template.

Windows Template

Take **Windows Server 2008 R2 Enterprise** and **Windows Server 2012 R2 Datacenter** for example, it may also apply to other windows version.

Note: Samba service must running to provision windows OS

- Import windows OS media

```
mount -o loop cn_windows_server_2008_r2_standard_enterprise_datacenter_and_web_with_
↪sp1_x64_dvd_617598.iso /media
rsync -az /media/ /opt/cloudboot/home/samba/windows/2008r2/
umount /media
```

Note:

- *Windows Server 2008 R2I* OS media is under folder `/opt/cloudboot/home/samba/windows/2008r2`
- Drivers is under folder `/opt/cloudboot/home/samba/windows/drivers/2008r2`. Using *model name* as sub-folder, and put driver's `driver.sys` and `driver.inf` under the sub-folder is recommended.
- `winconfig.exe` program under `/opt/cloudboot/home/samba/windows/firstboot` folder is installed by Cloudboot to update provision progress and configure windows OS including disk partitioning, network setting, user and registry setting, etc. User may upload `preinstall.cmd` and/or `postinstall.cmd` batch file to the same folder. CloudBoot will run these two file automatically at the right provision phase.

- Select `win2008r2-x86_64` as OS type, and `win2008r2-x86_64` as OS tempalte while create provision job

- The default administrator password is `yunjikeji`
- A sample of folder structure for windows OS drivers is shown as below:

```
/opt/cloudboot/home/samba/windows/drivers/
|-- 2008r2
|   |-- broadcom
|   |-- intel_10gb
|   |-- intel_40gb
|   |-- intel_pro100
|   |-- intel_pro1000
|   |-- kvm
|   |-- lsi_sas2
|   |-- lsi_sas3
|   |-- megasas2
|   |-- megasr1
|   `-- percsas3
`-- 2012r2
    |-- broadcom
    |-- intel_10gb
    |-- intel_40gb
    |-- intel_pro100
    |-- intel_pro1000
    |-- kvm
    |-- lsi_sas2
    |-- lsi_sas3
    |-- megasas2
    |-- megasr1
    `-- percsas3
```

XenServer Template

- Import XenServer 6.5 OS image

```
mount -o loop XenServer-6.5.0-xenserver.org-install-cd.iso /media/
rsync -az /media/ /opt/cloudboot/home/www/xenserver/6.5/
```

- Select `xenserver6.5-x86_64` as OS type, and `xenserver6.5` as OS tempalte while create provision job

CHAPTER 7

Virtual Environment Provision

Provision Virtual Host

In the following example, we create a lightweight KVM host environment.

Hardware Settings

- x86 server
- Support **Intel® Virtualization Technology** and be turned on in BIOS

System Settings

- RedHat/CentOS 6.0 or higher, with kvm support installed
- Firewall is off
- selinux is off
- Host and guest OS is in same subnet
- Create a seperated LVM pool for disk allocation
- CloudBoot server could ssh to host using certificate (instead of password)

Host Template

Cloudboot provides a sample templage named `centos6.7-kvmserver` for user referance, as shown below:

```
## create network bridge br0
# cat > /etc/sysconfig/network-scripts/ifcfg-br0 <<EOF
DEVICE=br0
BOOTPROTO=none
IPADDR=$IPADDR
NETMASK=$NETMASK
ONBOOT=yes
TYPE=Ethernet
NM_CONTROLLED=no
TYPE=Bridge
DELAY=0
EOF

# cat > /etc/sysconfig/network-scripts/ifcfg-eth0 <<EOF
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
TYPE=Ethernet
NM_CONTROLLED=no
TYPE=Ethernet
BRIDGE=br0
EOF

### create LVM Pool named guest_images_lvmedit /opt/cloudboot/etc/cloudboot-server/
↪cloudboot-server.conf to change its settings
virsh pool-define-as guest_images_lvm logical - - - VolGroup0 /dev/VolGroup0
virsh pool-autostart guest_images_lvm
virsh pool-start guest_images_lvm

### ssh channel from cloudboot server to host
test -f /opt/cloudboot/root/.ssh/id_rsa || ssh-keygen -t rsa -f /opt/cloudboot/root/.
↪ssh/id_rsa -C '' -N ''
chmod 600 /opt/cloudboot/root/.ssh/*
ssh-copy-id -i /opt/cloudboot/root/.ssh/id_rsa.pub [host_ip_address]
```

Virtual Machine Provision

- User could create virtual machine directly from CloudBoot UI after provision and refresh the host machine. It also support day 2 operation like start/stop/dicommission.
- While create a new virtual machine, user could specify hardware setting such as nubmer of CPU, memory, disk size, etc.
- OS provision settings for virtual machine is the same as the one for physical machines.

Note: CloudBoot use `MAC address` as virtual machine's serial number.

CHAPTER 8

CloudBoot API Referance

This chapter describes CloudBoot web service API with code examples.

User Related

Login

Request

Table 8.1: Request

Field	Description
URL	http://localhost:8083/api/osinstall/v1/user/login
encode	UTF-8
method	HTTP POST
payload	application/json

Payload

Table 8.2: Payload

Field	Type	Required	Description
username	string	yes	username
password	string	yes	password

Payload Sample

```
{
"Username": "admin",
"Password": "admin"
}
```

Code Sample (PHP)

```
<?php
    $data = array(
    "Username" => "admin",
    "Password" => "admin",
    );
    $str = json_encode($data);
    $ch = curl_init('http://localhost:8083/api/osinstall/v1/user/login');
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
    curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Content-Type: application/json',
        'Content-Length: ' . strlen($str))
    );

    $result = curl_exec($ch);
    echo curl_error($ch);
    echo $result;
?>
```

Response

Table 8.3: Response Format

Field	Description
Status	success or failure
Content	user infomation and access token
Message	return message

Sample Response Message

```
{
    "Content": {
        "ID": 1,
        "Username": "admin",
        "Name": "Super Administrator",
        "Role": "Administrator",
        "AccessToken": "097B55289D87C26FC33C2B0F7F80701D"
    },
    "Message": "Login Success",
    "Status": "success"
}
```

Logout

Request

Table 8.4: Request

Field	Description
URL	http://localhost:8083/api/osinstall/v1/user/logout
encode	UTF-8
method	HTTP POST
payload	application/json

Payload

Table 8.5: Payload

Field	Type	Required	Description
AccessToken	string	yes	access token

Payload Sample

```
{
    "AccessToken": "097B55289D87C26FC33C2B0F7F80701D"
}
```

Code Sample (PHP)

```
<?php
     $data = array(
     "AccessToken" => "097B55289D87C26FC33C2B0F7F80701D",
     );
     $str = json_encode($data);
     $ch = curl_init('http://localhost:8083/api/osinstall/v1/user/logout');
     curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
     curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
     curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
     curl_setopt($ch, CURLOPT_HTTPHEADER, array(
         'Content-Type: application/json',
         'Content-Length: ' . strlen($str))
     );

     $result = curl_exec($ch);
     echo curl_error($ch);
     echo $result;
 ?>
```

Response

Table 8.6: Response Format

Field	Description
Status	success or failure
Message	return message

Sample Response Message

```
{
    "Message": "User logout success",
    "Status": "success"
}
```

Data Import

Import Device

Request

Table 8.7: Request

Field	Description
URL	http://localhost:8083/api/osinstall/v1/device/add
encode	UTF-8
method	HTTP POST
payload	JSON

Payload

Table 8.8: Payload

Field	Type	Required	Description
Sn	string	yes	device serial number
Hostname	string	yes	Hostname
Ip	string	yes	ip address
ManageIp	string	no	management ip
NetworkID	int	yes	networks ID
ManageNetworkID	int	yes	management network ID
OsID	int	yes	os_configs ID
HardwareID	int	yes	hardwares ID
SystemID	int	yes	system_configs ID
LocationID	int	yes	location ID
AssetNumber	string	no	asset ID
AccessToken	string	yes	user access token

Payload Sample

```
{
    "Sn": "test",
    "Hostname": "idcos-test",
    "Ip": "192.168.0.3",
    "ManageIp": "192.168.1.1",
    "NetworkID": 6,
    "ManageNetworkID": 1,
    "OsID": 2,
    "HardwareID": 1,
    "SystemID": 1,
    "LocationID": 33,
    "AssetNumber": "CB20151216001",
    "AccessToken": "097B55289D87C26FC33C2B0F7F80701D",
}
```

Code Sample (PHP)

```
<?php
   $data = array(
    "Sn" => "test",
    "Hostname" => "idcos-test",
    "Ip" => "192.168.0.3",
    "ManageIp" => "192.168.1.1",
    "NetworkID" => 6,
    "ManageNetworkID" => 1,
    "OsID" => 2,
    "HardwareID" => 1,
    "SystemID" => 1,
    "LocationID" => 33,
    "AssetNumber" => "CB20151216001",
    "AccessToken" => "097B55289D87C26FC33C2B0F7F80701D",
    );
    $str = json_encode($data);
    $ch = curl_init('http://localhost:8083/api/osinstall/v1/device/add');
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
    curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Content-Type: application/json',
        'Content-Length: ' . strlen($str))
    );

    $result = curl_exec($ch);
    echo curl_error($ch);
    echo $result;
?>
```

Response

Table 8.9: Response Format

Field	Description
Status	success or failures
Message	return message

Sample Response Message

```
{
    "Message": "success",
    "Status": "success"
}
```

Provision Job Status Query

Request

Table 8.10: Request

Field	Description
URL	http://localhost:8083/api/osinstall/v1/device/isInInstallList
encode	UTF-8
method	HTTP POST
payload	application/json

Payload

Table 8.11: Payload

Field	Type	Required	Description
Sn	string	yes	device serial number

Payload Sample

```
{
    "Sn": "test"
}
```

Code Sample (PHP)

```
<?php
     $data = array("Sn" => "test");
     $str = json_encode($data);
```

```
    $ch = curl_init('http://localhost:8083/api/osinstall/v1/device/isInInstallList');
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
    curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Content-Type: application/json',
        'Content-Length: ' . strlen($str))
    );

    $result = curl_exec($ch);
    echo $result;
?>
```

Response

Table 8.12: Response Format

Field	Description
Status	success or failure
Content.Result	• `true` device is in provision queue • `false` device is NOT in provision queue
Message	return message

Sample Response Message

```
{
    "Content": {
        "Result": "true"
    },
    "Message": "Device is in provisioning queue",
    "Status": "success"
}
```

Log Update

Update Provision Job Progress

Note: This API is used by BootOS agent to update OS provisioning progress and logs.

Request

Table 8.13: Request

Field	Description
URL	http://localhost:8083/api/osinstall/v1/report/deviceInstallInfo
encode	UTF-8
method	HTTP POST
payload	application/json

Payload

Table 8.14: Payload

Field	Type	Required	Description
Sn	string	yes	device serial number
InstallProgress	float(112)	yes	• valus between 0~1 • -1 means provision job failed • 1 means provision job success
Title	string	no	message title
InstallLog	string	no	message body, requires `base64encode` encode

Payload Sample

```
{
   "Sn": "test",
   "InstallProgress": 0.1,
   "Title": "enter bootos",
   "InstallLog": "5byA5aeL6L+b5YWlYm9vdG9z"
}
```

Code Sample (PHP)

```
<?php
     $data = array("Sn" => "test","InstallProgress" => 0.1,"Title" => "enter bootos",
↪"InstallLog" => base64_encode("entering bootos"));
     $str = json_encode($data);
     $ch = curl_init('http://localhost:8083/api/osinstall/v1/report/deviceInstallInfo
↪');
     curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
     curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
     curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
     curl_setopt($ch, CURLOPT_HTTPHEADER, array(
         'Content-Type: application/json',
         'Content-Length: ' . strlen($str))
     );
```

```
    $result = curl_exec($ch);
    echo curl_error($ch);
    echo $result;
?>
```

Response

Table 8.15: Response Format

Field	Description
Status	success or failure
Content.Result	• `true` success • `false` failed
Message	return message

Sample Response Message

```
{
    "Content": {
        "Result": "true"
    },
    "Message": "success",
    "Status": "success"
}
```

Hardware Template Query

Request

Table 8.16: Request Format

Field	Description
URL	http://localhost:8083/api/osinstall/v1/device/getHardwareBySn
encode	UTF-8
method	HTTP POST
payload	application/json

Payload

Table 8.17: Payload

Field	Type	Required	Description
Sn	string	yes	serial number

Payload Sample

```
{
   "Sn": "test",
}
```

Code Sample (PHP)

```
<?php
  $data = array("Sn" => "test");
  $str = json_encode($data);
  $ch = curl_init('http://localhost:8083/api/osinstall/v1/device/getHardwareBySn');
  curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
  curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  curl_setopt($ch, CURLOPT_HTTPHEADER, array(
      'Content-Type: application/json',
      'Content-Length: ' . strlen($str))
  );

  $result = curl_exec($ch);
  echo curl_error($ch);
  echo $result;
  ?>
```

Response

Table 8.18: Response Format

Field	Description
Status	success or failure
Content.Compamy	vendor name
Content.Product	product name
Content.ModelName	module name
Content.Hardware	configure node
Content.Hardware[].Name	target name
Content.Hardware[].Scripts	script node
Content.Hardware[].Scripts[].script	script content, encoded by `base64decode`
Message	return message

Sample Response Message

```
{
    "Content": {
        "Company": "Dell",
        "Hardware": [
        {
            "Name": "RAID",
            "Scripts": [
            {
                "Name": "RAID",
```

```
                "Script": "L29wdC95dW5qaS9vc2luc3RhbGwvZGVsbC9yYWlkLnNoIC1jIC1sIDEw"
            }
            ]
        },
        {
            "Name": "OOB",
            "Scripts": [
            {
                "Name": "network type",
                "Script": "L29wdC95dW5qaS9vc2luc3RhbGwvZGVsbC9vb2Iuc2ggLW4gZGhjcA=="
            },
            {
                "Name": "username",
                "Script": "L29wdC95dW5qaS9vc2luc3RhbGwvZGVsbC9vb2Iuc2ggLXUgcm9vdA=="
            },
            {
                "Name": "passowrd",
                "Script": "L29wdC95dW5qaS9vc2luc3RhbGwvZGVsbC9vb2Iuc2ggLXAgY2Fsdmlu"
            }
            ]
        },
        {
            "Name": "BIOS",
            "Scripts": [
            {
                "Name": "VT",
                "Script":
↪"L29wdC95dW5qaS9vc2luc3RhbGwvZGVsbC9iaW9zLnNoIC10IGVuYWJsZQ=="
            },
            {
                "Name": "C-States",
                "Script":
↪"L29wdC95dW5qaS9vc2luc3RhbGwvZGVsbC9iaW9zLnNoIC1jIGRpc2FibGU="
            }
            ]
        }
        ],
        "ModelName": "R420",
        "Product": "PowerEdge"
    },
    "Message": "get hardware information success",
    "Status": "success"
}
```

PXE Query

Request

Table 8.19: Request Format

Field	Description
URL	http://localhost:8083/api/osinstall/v1/report/deviceMacInfo
encode	UTF-8
method	HTTP POST
payload	application/json

Payload

Table 8.20: Payload

Field	Type	Required	Description
Sn	string	yes	device serial number
MAC	string	yes	device MAC address

Payload Sample

```
{
   "Sn": "test",
   "Mac": "EA:1F:2d:3a:4H"
}
```

Code Sample (PHP)

```
<?php
  $data = array("Sn" => "test","Mac" => "EA:1F:2d:3a:4H");
  $str = json_encode($data);
  $ch = curl_init('http://localhost:8083/api/osinstall/v1/report/deviceMacInfo');
  curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
  curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  curl_setopt($ch, CURLOPT_HTTPHEADER, array(
      'Content-Type: application/json',
      'Content-Length: ' . strlen($str))
  );

  $result = curl_exec($ch);
  echo $result;
  ?>
```

Response

Table 8.21: Response Format

Field	Description
Status	success or failure
Content.Result	• `true` success • `false` failed
Message	return message

Sample Response Message

```
{
    "Content": {
        "Result": "true"
    },
    "Message": "success",
    "Status": "success"
}
```

OS Template Query

Request

Table 8.22: Request Format

Field	Description
URL	http://localhost:8083/api/osinstall/v1/device/getSystemBySn
encode	UTF-8
method	HTTP GET
payload	text/html

Payload

Table 8.23: Payload

Field	Type	Required	Description
sn	string	yes	device serial number
type	string	yes	specify result format, could be `json` or `raw`, default is `raw`

Request Sample

```
http://localhost:8083/api/osinstall/v1/device/getSystemBySn?sn=test&type=raw
```

Code Sample (PHP)

```
<?php
    $url = "http://localhost:8083/api/osinstall/v1/device/getSystemBySn?sn=test&
↪type=raw";
    $content = file_get_contents($url);
    echo $content;
?>
```

Sample Response Message

```
install
url --url=http://mirror.idcos.com/centos/6.7/os/x86_64/
lang en_US.UTF-8
keyboard us
network --onboot yes --device bootif --bootproto dhcp --noipv6
rootpw  --iscrypted $6$eAdCfx9hZjVMqyS6
↪$BYIbEu4zeKp0KLnz8rLMdU7sQ5o4hQRv55o151iLX7s2kSq.5RVsteGWJlpPMqIRJ8.
↪WUcbZC3duqX0Rt3unK/
firewall --disabled
authconfig --enableshadow --passalgo=sha512
selinux --disabled
timezone Asia/Shanghai
text
reboot
zerombr
bootloader --location=mbr --append="console=tty0 biosdevname=0 audit=0 selinux=0"
clearpart --all --initlabel
part /boot --fstype=ext4 --size=256 --ondisk=sda
part swap --size=2048 --ondisk=sda
part / --fstype=ext4 --size=100 --grow --ondisk=sda

%packages --ignoremissing
@base
@core
@development

%pre
_sn=$(dmidecode -s system-serial-number 2>/dev/null | awk '/^[^#]/ { print $1 }')
curl -H "Content-Type: application/json" -X POST -d "{\"Sn\":\"$_sn\",\"Title\":\
↪"start os provisioning\",\"InstallProgress\":0.6,\"InstallLog\":\"SW5zdGFsbCBPUwo=\
↪"}" http://osinstall.idcos.com/api/osinstall/v1/report/deviceInstallInfo
curl -H "Content-Type: application/json" -X POST -d "{\"Sn\":\"$_sn\",\"Title\":\
↪"disk partition and software package installation\",\"InstallProgress\":0.7,\
↪"InstallLog\":\"SW5zdGFsbCBPUwo=\"}" http://osinstall.idcos.com/api/osinstall/v1/
↪report/deviceInstallInfo

%post
progress() {
    curl -H "Content-Type: application/json" -X POST -d "{\"Sn\":\"$_sn\",\"Title\":\"
↪$1\",\"InstallProgress\":$2,\"InstallLog\":\"$3\"}" http://osinstall.idcos.com/api/
↪osinstall/v1/report/deviceInstallInfo
}

_sn=$(dmidecode -s system-serial-number 2>/dev/null | awk '/^[^#]/ { print $1 }')
```

```
progress "set hostname and networking" 0.8 "Y29uZmlnIG5ldHdvcmsK"

# config network
cat > /etc/modprobe.d/disable_ipv6.conf <<EOF
install ipv6 /bin/true
EOF

curl -o /tmp/networkinfo "http://osinstall.idcos.com/api/osinstall/v1/device/
↪getNetworkBySn?sn=${_sn}&type=raw"
source /tmp/networkinfo

cat > /etc/sysconfig/network <<EOF
NETWORKING=yes
HOSTNAME=$HOSTNAME
GATEWAY=$GATEWAY
NOZEROCONF=yes
NETWORKING_IPV6=no
IPV6INIT=no
PEERNTP=no
EOF

cat > /etc/sysconfig/network-scripts/ifcfg-eth0 <<EOF
DEVICE=eth0
BOOTPROTO=static
IPADDR=$IPADDR
NETMASK=$NETMASK
ONBOOT=yes
TYPE=Ethernet
NM_CONTROLLED=no
EOF

progress "add user" 0.85 "YWRkIHVzZXIgeXVuamkK"
useradd yunji

progress "configure system service" 0.9 "Y29uZmlnIHN5c3RlbSBzZXJ2aWNlCg=="

# config service
service=(crond network ntpd rsyslog sshd sysstat)
chkconfig --list | awk '{ print $1 }' | xargs -n1 -I@ chkconfig @ off
echo ${service[@]} | xargs -n1 | xargs -I@ chkconfig @ on

progress "configure system parameter" 0.95 "Y29uZmlnIGJhc2ggcHJvbXB0Cg=="

# custom bash prompt
cat >> /etc/profile <<'EOF'

export LANG=en_US.UTF8
export PS1='\n\e[1;37m[\e[m\e[1;32m\u\e[m\e[1;33m@\e[m\e[1;
↪35m\H\e[m:\e[4m`pwd`\e[m\e[1;37m]\e[m\e[1;36m\e[m\n\$ '
export HISTTIMEFORMAT='[%F %T] '
EOF

progress "finish" 1 "aW5zdGFsbCBmaW5pc2hlZAo="
```

Device Networking Query

Request

Table 8.24: Request Format

Field	Description
URL	http://localhost:8083/api/osinstall/v1/device/getNetworkBySn
encode	UTF-8
method	HTTP GET
payload	text/html

Payload

Table 8.25: Payload

Field	Type	Required	Description
sn	string	yes	device serial number
type	string	yes	specify result format, could be `json` or `raw`, default is `raw`

Request Sample

```
http://localhost:8083/api/osinstall/v1/device/getNetworkBySn?sn=test&type=raw
```

Code Sample (PHP)

```php
<?php
    $url = "http://localhost:8083/api/osinstall/v1/device/getNetworkBySn?sn=test&
↪type=raw";
    $content = file_get_contents($url);
    echo $content;
?>
```

Sample Response Message

```
HOSTNAME="idcos-test"
IPADDR="192.168.0.3"
NETMASK="255.255.255.0"
GATEWAY="192.168.0.1"
VLAN="0"
Trunk="no"
Bonding="no"
HWADDR="53:54:00:99:2D:7C"
```

CHAPTER 9

Appendix

Cloudboot Version History

- 2015-12-16 first release
- 2016-03-02 add VMware ESXi and Windows support
- 2016-04-25 RPM-based installation, add support for unbuntu provisioning
- 2016-06-30 add XenServer supportadd virtual env management functionsss
- 2017-01-18 add hardware toolset function, support multi RAID group