Control Center Installation Guide

Release 1.5.0

Zenoss, Inc.

www.zenoss.com

Control Center Installation Guide

Copyright © 2017 Zenoss, Inc. All rights reserved.

Zenoss, Own IT, and the Zenoss logo are trademarks or registered trademarks of Zenoss, Inc., in the United States and other countries. All other trademarks, logos, and service marks are the property of Zenoss or other third parties. Use of these marks is prohibited without the express written consent of Zenoss, Inc., or the third-party owner.

Linux is a registered trademark of Linus Torvalds.

All other companies and products mentioned are trademarks and property of their respective owners.

Part Number: 1920.17.349

Zenoss, Inc.
11305 Four Points Drive
Bldg 1 - Suite 300
Austin, Texas 78726

Contents

About this guide

Control Center Installation Guide provides detailed procedures for installing and configuring a Control Center deployment. Before using this guide, carefully review the *Control Center Planning Guide*.

Zenoss customers: This guide does not include procedures for installing a high-availability deployment. For more information, refer to the *Control Center Installation Guide for High-Availability Deployments*.

Supported clients and browsers

The following table identifies the supported combinations of client operating systems and web browsers.

Client OS	Tested browsers
Windows 7, 10	Internet Explorer 11*
	Firefox 56 and later
	Chrome 61 and later
macOS 10.12.3, 10.13	Firefox 56 and later
	Chrome 61 and later
Ubuntu 14.04 LTS	Firefox 56 and later
	Chrome 61 and later

Related publications

Title	Description
Control Center Release Notes	Describes known issues, fixed issues, and late-breaking information not included in other publications.
Control Center Planning Guide	Provides both general and specific information about preparing to deploy Control Center.
Control Center Installation Guide	Provides detailed procedures for installing and configuring Control Center.
Control Center Installation Guide for High-Availability Deployments	Provides detailed procedures for installing and configuring Control Center in a high-availability deployment.
Control Center Reference Guide	Provides information and procedures for managing Control Center. This information is also available as online help in the Control Center browser interface.
Control Center Upgrade Guide	Provides detailed procedures for updating a Control Center deployment to the latest release.
Control Center Upgrade Guide for High-Availability Deployments	Provides detailed procedures for updating a high-availability deployment of Control Center to the latest release.

* Enterprise mode only; compatibility mode is not tested.

Documentation feedback

To provide feedback about this document, or to report an error or omission, please send an email to `docs@controlcenter.io`. In the email, please include the document title (*Control Center Installation Guide*) and part number (1920.17.349) and as much information as possible about the context of your feedback.

Change history

The following list associates document part numbers and the important changes to this guide since the previous release. Some of the changes involve features or content, but others do not. For information about new or changed features, refer to the *Control Center Release Notes*.

1920.17.349 (1.5.0)

Specify Docker version in `yum install` command.

Reinstate steps to disable firewall.

1920.17.331 (1.5.0)

Replace Leapfile.net with delivery.zenoss.com.

Add steps for importing the Zenoss GPG key.

1920.17.311 (1.5.0)

Replace Docker CE 17.03.1 with 17.09.0.

Replace RHEL/CentOS 7.1 with 7.4.

1320.17.268

Update release number (1.4.1).

1320.17.242

Correct a previous modification for configuring Docker. The `--insecure-registry` flag must be set when the value of *SERVICED_DOCKER_REGISTRY* is not `localhost:5000`.

1320.17.187

Modify the `yum` command used to install Docker CE 17.03.1.

1320.17.171

Update release number (1.3.3).

Add Docker CE 17.03.1.

Add ZooKeeper variables for tuning TCP/IP communications with resource pools.

Remove step for disabling SELinux.

1320.17.122

Update release number (1.3.2).

1320.17.100

Update release number (1.3.1).

Modify the step for configuring Docker on master hosts. The `--insecure-registry` flag is only needed on delegate hosts.

1320.17.076

Update release number (1.3.0).

Add clarification that offline installations must be performed completely offline.

Add procedure for adding delegates when using network address translation (NAT).

1320.17.59

Update release number (1.2.3).

Remove `TLS_RSA_WITH_RC4_128_SHA` from list of ciphers associated with *SERVICED_TLS_CIPHERS*.

1320.17.024

Update release number (1.2.2).

1320.16.351

Minor changes to synchronize with the Upgrade Guide.

1320.16.350

Add steps to download and use the `serviced` RPM file (Zenoss customers only).

Add procedures for importing the ZooKeeper image file on offline nodes (Zenoss customers only).

Update release number (1.2.1).

1320.16.327

Add new best practice guideline for managing configuration files.

1320.16.319

Initial release (1.2.0).

1 Downloading and staging required files

This chapter describes how to download and install or stage Control Center software and its operating system dependencies. The procedures in this chapter are required to perform an installation.

The following table identifies where to perform each procedure in this chapter.

Procedure	Where to perform
Downloading Control Center files on page 8	A workstation with internet access
Installing the repository mirror on page 9	All Control Center hosts
Staging Docker image files on page 11	The Control Center master host
Staging a Docker image file on ZooKeeper ensemble nodes on page 11	Delegate hosts that are ZooKeeper ensemble nodes

Downloading Control Center files

To perform this procedure, you need:

- A workstation with internet access.
- **Zenoss Resource Manager users:** Permission to download files from *delivery.zenoss.com*. Customers can request permission by filing a ticket at the *Zenoss Support* site.
- **Zenoss Core users:** An account on the *Zenoss Community* site.
- A secure network copy program.

Use this procedure to

- download the required files to a workstation
- copy the files to the hosts that need them

Perform these steps:

1. In a web browser, navigate to the download site, and then log in.

   Zenoss Resource Manager users: *delivery.zenoss.com*
   Zenoss Core users: *Zenoss Community*
2. Download the self-installing Docker image files.

   ```
   install-zenoss-serviced-isvcs-v61.run
   ```

```
install-zenoss-isvcs-zookeeper-v10.run
```

3 Download the Control Center RPM file.

```
serviced-1.5.0-1.x86_64.rpm
```

4 Identify the operating system release on Control Center hosts.

Enter the following command on each Control Center host in your deployment, if necessary. All Control Center hosts should be running the same operating system release and kernel.

```
cat /etc/redhat-release
```

5 Download the RHEL/CentOS repository mirror file for your deployment.

The download site provides a repository mirror file for each tested release of RHEL/CentOS. Each mirror file contains the release-specific packages that Control Center requires.

```
yum-mirror-centos7.2-1511-serviced-1.5.0.x86_64.rpm
yum-mirror-centos7.3-1611-serviced-1.5.0.x86_64.rpm
yum-mirror-centos7.4-1708-serviced-1.5.0.x86_64.rpm
```

6 Optional: Download the Zenoss GNU Privacy Guard (GPG) key, if desired.

You can use the Zenoss GPG key to verify Zenoss RPM files and the `yum` metadata of the repository mirror.

a Download the key.

```
curl --location -o /tmp/tmp.html \
'http://keys.gnupg.net/pks/lookup?op=get&search=0xED0A5FD2AA5A1AD7'
```

b Determine whether the download succeeded.

```
grep -Ec '^\-\-\-\-\-BEGIN PGP' /tmp/tmp.html
```

- If the result is `0`, return to the previous substep.
- If the result is `1`, proceed to the next substep.

c Extract the key.

```
awk '/^-----BEGIN PGP.*$/,/^-----END PGP.*$/' \
  /tmp/tmp.html > ./RPM-GPG-KEY-Zenoss
```

7 Use a secure copy program to copy the files to Control Center hosts.

- Copy all files to the master host.
- Copy the RHEL/CentOS RPM file, the Control Center RPM file, and the Zenoss GPG key file to all delegate hosts.
- Copy the Docker image file for ZooKeeper to delegate hosts that are ZooKeeper ensemble nodes.

Installing the repository mirror

Use this procedure to install the Zenoss repository mirror on a Control Center host. The mirror contains packages that are required on all Control Center hosts.

1 Log in to the target host as `root`, or as a user with superuser privileges.
2 Move the RPM files and the Zenoss GPG key file to `/tmp`.
3 Install the repository mirror.

```
yum install /tmp/yum-mirror-*.rpm
```

The `yum` command copies the contents of the RPM file to `/opt/zenoss-repo-mirror`.

4. Optional: Install the Zenoss GPG key, and then test the package files, if desired.
   a. Move the Zenoss GPG key to the mirror directory.

   ```
   mv /tmp/RPM-GPG-KEY-Zenoss /opt/zenoss-repo-mirror
   ```

   b. Install the key.

   ```
   rpm --import /opt/zenoss-repo-mirror/RPM-GPG-KEY-Zenoss
   ```

   c. Test the repository mirror package file.

   ```
   rpm -K /tmp/yum-mirror-*.rpm
   ```

   On success, the result includes the file name and the following information:

   ```
   (sha1) dsa sha1 md5 gpg OK
   ```

   d. Test the Control Center package file.

   ```
   rpm -K /tmp/serviced-1.5.0-1.x86_64.rpm
   ```

5. Optional: Update the configuration file of the Zenoss repository mirror to enable GPG key verification, if desired.
   a. Open the repository mirror configuration file (`/etc/yum.repos.d/zenoss-mirror.repo`) with a text editor, and then add the following lines to the end of the file.

   ```
   repo_gpgcheck=1
   gpgkey=file:///opt/zenoss-repo-mirror/RPM-GPG-KEY-Zenoss
   ```

   b. Save the file, and then close the editor.
   c. Update the `yum` metadata cache.

   ```
   yum makecache fast
   ```

   The cache update process includes the following prompt:

   ```
   Retrieving key from file:///opt/zenoss-repo-mirror/RPM-GPG-KEY-
   Zenoss
   Importing GPG key 0xAA5A1AD7:
    Userid     : "Zenoss, Inc. <dev@zenoss.com>"
    Fingerprint: f31f fd84 6a23 b3d5 981d a728 ed0a 5fd2 aa5a 1ad7
    From       : /opt/zenoss-repo-mirror/RPM-GPG-KEY-Zenoss
   Is this ok [y/N]:
   ```

   Enter `y`.

6. Move the Control Center package file to the mirror directory.

   ```
   mv /tmp/serviced-1.5.0-1.x86_64.rpm /opt/zenoss-repo-mirror
   ```

7. Optional: Delete the mirror package file, if desired.

   ```
   rm /tmp/yum-mirror-*.rpm
   ```

Staging Docker image files

Before performing this procedure, verify that approximately 640MB of temporary space is available on the file system where `/root` is located.

Use this procedure to copy Docker image files to a Control Center host. The files are used when Docker is fully configured.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Copy or move the archive files to `/root`.
3. Add execute permission to the files.

```
chmod +x /root/*.run
```

Staging a Docker image file on ZooKeeper ensemble nodes

Before performing this procedure, verify that approximately 170MB of temporary space is available on the file system where `/root` is located.

Use this procedure to add a Docker image file to the Control Center delegate hosts that are ZooKeeper ensemble nodes. Delegate hosts that are not ZooKeeper ensemble nodes do not need the file.

1. Log in to a delegate host as `root`, or as a user with superuser privileges.
2. Copy or move the `install-zenoss-isvcs-zookeeper-v10.run` file to `/root`.
3. Add execute permission to the file.

```
chmod +x /root/*.run
```

Installing a master host 2

This chapter describes how to install Control Center on a Red Hat Enterprise Linux (RHEL) or CentOS host. The candidate host must have the CPU, RAM, and storage resources required to serve as the Control Center master host.

For more information about master host requirements, refer to the *Control Center Planning Guide*.

Verifying candidate host resources

Use this procedure to determine whether the hardware resources and installed operating system of a host are sufficient to serve as a Control Center master host.

1. Log in to the candidate host as `root`, or as a user with superuser privileges.
2. Verify that the host implements the 64-bit version of the x86 instruction set.

   ```
   uname -m
   ```

   - If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step
   - If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.
3. Verify that the host has adequate storage space for Docker temporary files and audit logging.
   a. Display the amount of space available in `/tmp`.

      Docker requires 10GB of storage for temporary files, and the installation process includes instructions to link the Docker temporary directory to `/tmp`.

      ```
      df -h /tmp
      ```

      Typically, `/tmp` is mounted on the root filesystem, `/`.
   b. Display the amount of space available in `/var/log`.

      By default, Control Center requires 10GB of space for audit logging in `/var/log`. The amount of space required for audit logs, and their location, is configurable. For more information, refer to the *Control Center Reference Guide*.

      ```
      df -h /var/log
      ```

      Like `/tmp`, `/var/log` is typically mounted on the root filesystem, `/`.

If the result does not include a minimum of 10GB of space for Docker temporary files and 10GB for audit logging, stop this procedure and select a different host.

4 Determine whether the available memory and swap is sufficient.

   a Display the available memory.

   ```
   free -h
   ```

   b Compare the available memory and swap space with the amount required for a master host in your deployment.

   For more information, refer to the *Control Center Planning Guide*.

If the result does not meet minimum requirements, stop this procedure and select a different host.

5 Determine whether the CPU resources are sufficient.

   a Display the total number of CPU cores.

   ```
   cat /proc/cpuinfo | grep -Ec '^core id'
   ```

   b Compare the available resources with the requirements for a Control Center master host.

   For more information, refer to the *Control Center Planning Guide*.

6 Determine whether the CPU resources support the AES instruction set.

```
cat /proc/cpuinfo | grep -Ec '^flags.*aes'
```

For optimal performance, the result of the preceding commands must match the total number of CPU resources available on the host. If the result is 0, performance is severely degraded.

If the result is 0 and the candidate host is a virtual machine, the managing hypervisor may be configured in Hyper-V compatibility mode. Check the setting and disable it, if possible, or select a different host.

7 Determine whether the installed operating system release is one of the releases that has been tested with Control Center.

```
cat /etc/redhat-release
```

- If the result includes `7.2`, `7.3`, or `7.4` proceed to the next step.
- If the result does not include `7.2`, `7.3`, or `7.4`, select a different host, and then start this procedure again.

8 Ensure the host has a persistent numeric ID.

Skip this step if you are installing a single-host deployment.

Each Control Center host must have a unique host ID, and the ID must be persistent (not change when the host reboots).

```
test -f /etc/hostid || genhostid ; hostid
```

Record the ID for comparison with other Control Center hosts.

9 Verify that name resolution works on this host.

```
hostname -i
```

If the result is not a valid IPv4 address, add an entry for the host to the network nameserver, or to `/etc/hosts`.

10 Add an entry to `/etc/hosts` for localhost, if necessary.

a. Determine whether `127.0.0.1` is mapped to `localhost`.

```
grep 127.0.0.1 /etc/hosts | grep localhost
```

If the preceding commands return no result, perform the following substep.

b. Add an entry to `/etc/hosts` for `localhost`.

```
echo "127.0.0.1 localhost" >> /etc/hosts
```

11. Update the Linux kernel, if necessary.

a. Determine which kernel version is installed.

```
uname -r
```

If the result is lower than `3.10.0-327.22.2.el7.x86_64`, perform the following substep.

b. Update the kernel, and then restart the host.

The following commands require internet access or a local mirror of operating system packages.

```
yum makecache fast && yum update kernel && reboot
```

Master host storage requirements

This section provides a quick reference for the minimum storage requirements of a Control Center master host. These requirements are in addition to the storage required for the host's operating system. For more information, refer to the *Control Center Planning Guide*.

Control Center hosts need either unformatted block storage devices or partitions, or free space in one or more LVM volume groups.

- Enter the following command to display information about block storage:

```
lsblk -ap --output=NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT
```

- Enter the following command to display information about LVM volume groups:

```
vgdisplay
```

Compare the output with the information in the following table.

Purpose	Minimum size	Description
Docker data	50GB (required)	Local, high-performance storage.
Control Center internal services data	50GB (required)	Local, high-performance storage formatted as an XFS filesystem.
Application data	200GB (suggested)	Local, high-performance storage.
Application data backups	150GB (suggested)	A remote file server compatible with XFS or a local storage formatted as an XFS filesystem. If the storage is local, ensure that it does not cause contention with the storage for Control Center internal services.

The storage for Docker data and application data is configured as LVM thin pools in subsequent procedures.

The suggested minimum sizes for application data and application data backups should be replaced with sizes that meet your application requirements. To calculate the appropriate sizes for these storage areas, use the following guidelines:

- Application data storage includes space for both data and snapshots. The default base size for data is 100GB, and the recommended space for snapshots is 100% of the base size. Adding the two yields the suggested minimum size of 200GB.
- For application data backups, the recommended space is 150% of the base size for data.

The following sections provide procedures for formatting and mounting local storage as XFS filesystems for Control Center internal services data and application data backups.

Creating a filesystem for Control Center internal services

This procedure requires one unused device or partition.

Use this procedure to create an XFS filesystem on an unused device or partition.

1. Log in to the target host as `root`, or as a user with superuser privileges.
2. Identify the target device or partition for the filesystem to create.

```
lsblk -ap --output=NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT
```

3. Create an XFS filesystem.
   Replace *Storage* with the path of the target device or partition:

```
mkfs.xfs Storage
```

4. Enter the following command to add an entry to the `/etc/fstab` file.
   Replace *Storage* with the path of the device or partition used in the previous step:

```
echo "Storage /opt/serviced/var/isvcs xfs defaults 0 0" >> /etc/fstab
```

5. Create the mount point for internal services data.

```
mkdir -p /opt/serviced/var/isvcs
```

6. Mount the filesystem, and then verify it mounted correctly.

```
mount -a && mount | grep isvcs
```

   Example result:

```
/dev/xvdb1 on /opt/serviced/var/isvcs type xfs
 (rw,relatime,attr2,inode64,noquota)
```

Creating a filesystem for application data backups

This procedure requires one unused device or partition, or a remote file server that is compatible with XFS.

Use this procedure create an XFS filesystem on a device or partition, or to mount a remote filesystem, for application data backups.

Note If you are using a partition on a local device for backups, ensure that the storage for Control Center internal services data is not on the same device.

1. Log in to the target host as `root`, or as a user with superuser privileges.
2. Optional: Identify the target device or partition for the filesystem to create, if necessary.

   Skip this step if you are using a remote file server.

   ```
   lsblk -ap --output=NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT
   ```
3. Optional: Create an XFS filesystem, if necessary.

   Skip this step if you are using a remote file server.

   Replace *Storage* with the path of the target device or partition:

   ```
   mkfs.xfs Storage
   ```
4. Create an entry in the `/etc/fstab` file.

   Replace *File-System-Specification* with one of the following values:

   - the path of the device or partition used in the previous step
   - the remote server specification

   ```
   echo "File-System-Specification \
     /opt/serviced/var/backups xfs defaults 0 0" >> /etc/fstab
   ```
5. Create the mount point for backup data.

   ```
   mkdir -p /opt/serviced/var/backups
   ```
6. Mount the filesystem, and then verify it mounted correctly.

   ```
   mount -a && mount | grep backups
   ```

   Example result:

   ```
   /dev/sdb3 on /opt/serviced/var/backups type xfs
    (rw,relatime,seclabel,attr2,inode64,noquota)
   ```

Preparing the master host operating system

Perform the steps in *Downloading and staging required files* on page 8, before performing this procedure.

Use this procedure to prepare a RHEL/CentOS host as a Control Center master host.

1. Log in to the candidate master host as `root`, or as a user with superuser privileges.
2. Disable the firewall, if necessary.

   This step is required for installation but not for deployment. For more information, refer to the *Control Center Planning Guide*.

   a. Determine whether the `firewalld` service is enabled.

   ```
   systemctl status firewalld.service
   ```

   - If the result includes `Active: inactive (dead)`, the service is disabled. Proceed to the next step.
   - If the result includes `Active: active (running)`, the service is enabled. Perform the following substep.

b Disable the `firewalld` service.

```
systemctl stop firewalld && systemctl disable firewalld
```

On success, the preceding commands display messages similar to the following example:

```
rm '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
rm '/etc/systemd/system/basic.target.wants/firewalld.service'
```

3 Optional: Enable persistent storage for log files, if desired.

By default, RHEL/CentOS systems store log data only in memory or in a ring buffer in the `/run/log/journal` directory. By performing this step, log data persists and can be saved indefinitely, if you implement log file rotation practices. For more information, refer to your operating system documentation.

Note The following commands are safe when performed during an installation, before Docker or Control Center are installed or running. To enable persistent log files after installation, stop Control Center, stop Docker, and then enter the following commands.

```
mkdir -p /var/log/journal && systemctl restart systemd-journald
```

4 Enable and start the Dnsmasq package.

The package facilitates networking among Docker containers.

```
systemctl enable dnsmasq && systemctl start dnsmasq
```

If name resolution in your environment relies solely on entries in `/etc/hosts`, configure `dsnmasq` so that containers can use the file:

a Open `/etc/dnsmasq.conf` with a text editor.

b Locate the line that starts with `#domain-needed`, and then make a copy of the line, immediately below the original.

c Remove the number sign character (`#`) from the beginning of the line.

d Locate the line that starts with `#bogus-priv`, and then make a copy of the line, immediately below the original.

e Remove the number sign character (`#`) from the beginning of the line.

f Locate the line that starts with `#local=/localnet/`, and then make a copy of the line, immediately below the original.

g Remove `net`, and then remove the number sign character (`#`) from the beginning of the line.

h Locate the line that starts with `#domain=example.com`, and then make a copy of the line, immediately below the original.

i Replace `example.com` with `local`, and then remove the number sign character (`#`) from the beginning of the line.

j Save the file, and then close the editor.

k Restart the `dnsmasq` service.

```
systemctl restart dnsmasq
```

5 Install and configure the NTP package.

Note This procedure assumes the host has internet access. To install and configure NTP on a host that does not have internet access, see *Configuring a private master NTP server* on page 86.

a Install the package.

```
yum install ntp
```

b Set the system time.

```
ntpd -gq
```

c Enable the `ntpd` daemon.

```
systemctl enable ntpd
```

d Configure `ntpd` to start when the system starts.

Currently, an unresolved issue associated with NTP prevents `ntpd` from restarting correctly after a reboot. The following commands provide a workaround to ensure that it does.

```
echo "systemctl start ntpd" >> /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
```

Installing Docker CE and Control Center

Perform the steps in *Downloading and staging required files* on page 8 before performing this procedure.

Use this procedure to install Docker CE and Control Center on a host.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Install Docker CE 17.09.0 from the local repository mirror.
   a Install Docker CE.

   ```
   yum install --enablerepo=zenoss-mirror docker-ce-17.09.0.ce
   ```

   If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 89 for potential resolutions.
   b Enable automatic startup.

   ```
   systemctl enable docker
   ```
3. Install Control Center 1.5.0 from the local repository mirror.
   a Install Control Center.

   ```
   yum install --enablerepo=zenoss-mirror \
     /opt/zenoss-repo-mirror/serviced-1.5.0-1.x86_64.rpm
   ```

   If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 89 for potential resolutions.
   b Enable automatic startup.

   ```
   systemctl enable serviced
   ```
4. Make a backup copy of the Control Center configuration file.
   a Make a copy of `/etc/default/serviced`.

   ```
   cp /etc/default/serviced /etc/default/serviced-1.5.0-orig
   ```

b Set the backup file permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.5.0-orig
```

Configuring Docker

Use this procedure to configure Docker.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Create a symbolic link for the Docker temporary directory.

   Docker uses its temporary directory to spool images. The default directory is `/var/lib/docker/tmp`. The following command specifies the same directory that Control Center uses, `/tmp`. You can specify any directory that has a minimum of 10GB of unused space.

   a Create the `docker` directory in `/var/lib`.

   ```
   mkdir /var/lib/docker
   ```

   b Create the link to `/tmp`.

   ```
   ln -s /tmp /var/lib/docker/tmp
   ```

3. Create a `systemd` drop-in file for Docker.

   a Create the override directory.

   ```
   mkdir -p /etc/systemd/system/docker.service.d
   ```

   b Create the unit drop-in file.

   ```
   cat <<EOF > /etc/systemd/system/docker.service.d/docker.conf
   [Service]
   TimeoutSec=300
   EnvironmentFile=-/etc/sysconfig/docker
   ExecStart=
   ExecStart=/usr/bin/dockerd \$OPTIONS
   TasksMax=infinity
   EOF
   ```

   c Reload the `systemd` manager configuration.

   ```
   systemctl daemon-reload
   ```

4. Create an LVM thin pool for Docker data.

   For more information about the `serviced-storage` command, see *serviced-storage* on page 71.

   To use an entire block device or partition for the thin pool, replace *Device-Path* with the device path:

   ```
   serviced-storage create-thin-pool docker Device-Path
   ```

   To use 50GB of an LVM volume group for the thin pool, replace *Volume-Group* with the name of an LVM volume group:

   ```
   serviced-storage create-thin-pool --size=50G docker Volume-Group
   ```

   On success, the result is the device mapper name of the thin pool, which always starts with `/dev/mapper`.
5. Configure and start the Docker service.

a Create a variable for the name of the Docker thin pool.

Replace *Thin-Pool-Device* with the name of the thin pool device created in the previous step:

```
myPool="Thin-Pool-Device"
```

b Create variables for adding arguments to the Docker configuration file. The `--exec-opt` argument is a workaround for *a Docker issue* on RHEL/CentOS 7.x systems.

```
myDriver="--storage-driver devicemapper"
myLog="--log-level=error"
myFix="--exec-opt native.cgroupdriver=cgroupfs"
myMount="--storage-opt dm.mountopt=discard"
myFlag="--storage-opt dm.thinpooldev=$myPool"
```

c Add the arguments to the Docker configuration file.

```
echo 'OPTIONS="'$myLog $myDriver $myFix $myMount $myFlag'"' \
  >> /etc/sysconfig/docker
```

d Start or restart Docker.

```
systemctl restart docker
```

The startup may take up to a minute, and may fail. If startup fails, repeat the `restart` command.

6 Configure name resolution in containers.

Each time it starts, `docker` selects an IPv4 subnet for its virtual Ethernet bridge. The selection can change; this step ensures consistency.

a Identify the IPv4 subnet and netmask `docker` has selected for its virtual Ethernet bridge.

```
ip addr show docker0 | grep inet
```

b Open `/etc/sysconfig/docker` in a text editor.

c Add the following flags to the end of the *OPTIONS* declaration.

Replace *Bridge-Subnet* with the IPv4 subnet `docker` selected for its virtual bridge:

```
--dns=Bridge-Subnet --bip=Bridge-Subnet/16
```

For example, if the bridge subnet is 172.17.0.1, add the following flags:

```
--dns=172.17.0.1 --bip=172.17.0.1/16
```

Note Use a space character ( ) to separate flags, and make sure the double quote character (") delimits the declaration of *OPTIONS*.

d Save the file, and then close the editor.

e Restart the Docker service.

```
systemctl restart docker
```

Loading image files

Use this procedure to load images into the local Docker registry on a host.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Change directory to `/root`.

```
cd /root
```

3. Load the images.

```
for image in install-zenoss-*.run
do
  /bin/echo -en "\nLoading $image..."
  yes | ./$image
done
```

4. List the images in the registry.

```
docker images
```

   The result should show one image for each archive file.
5. Optional: Delete the archive files, if desired.

```
rm -i ./install-zenoss-*.run
```

Creating the application data thin pool

Use this procedure to create a thin pool for application data storage.

The following Control Center configuration variables are used to specify application data thin pools:

SERVICED_FS_TYPE

Default: `devicemapper`

The driver to manage application data storage on the `serviced` master host. Only `devicemapper` is supported in production deployments.

The only supported storage layout for the `devicemapper` driver is an LVM thin pool. To create a thin pool, use the `serviced-storage` utility. To specify the name of the thin pool device, use the *SERVICED_DM_THINPOOLDEV* variable.

SERVICED_DM_THINPOOLDEV

Default: (none)

The name of the thin pool device to use with the `devicemapper` storage driver.

Note This procedure does not include a specific value for the size of the thin pool. For more information about sizing this resource, refer to the documentation for your application. Or, use the suggested minimum value, 200GB. You can add storage to an LVM thin pool at any time.

Perform these steps:

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Create an LVM thin pool for application data.

   For more information about the `serviced-storage` command, see *serviced-storage* on page 71.

   To use an entire block device or partition for the thin pool, replace *Device-Path* with the device path:

```
serviced-storage create-thin-pool serviced Device-Path
```

To use 200GB of an LVM volume group for the thin pool, replace *Volume-Group* with the name of an LVM volume group:

```
serviced-storage create-thin-pool --size=200G serviced Volume-Group
```

On success, the result is the device mapper name of the thin pool, which always starts with `/dev/mapper`. Record the name for use in the next step.

3 Edit storage variables in the Control Center configuration file.

   a Open `/etc/default/serviced` in a text editor.
   b Locate the line for the *SERVICED_FS_TYPE* variable, and then make a copy of the line, immediately below the original.
   c Remove the number sign character (`#`) from the beginning of the line.
   d Locate the line for the *SERVICED_DM_THINPOOLDEV* variable, and then make a copy of the line, immediately below the original.
   e Remove the number sign character (`#`) from the beginning of the line.
   f Set the value to the device mapper name of the thin pool for application data.
   g Save the file, and then close the editor.

Proceed to the next chapter and configure the host.

3 Configuring and starting the master host

This chapter includes the procedures for configuring Control Center on the master host, describes the configuration options that apply to the master host, and includes steps for starting the master host for the first time. Before installing Control Center on delegate hosts, perform the procedures in this chapter.

Many configuration choices depend on application requirements. Before configuring Control Center on the master host, review your application documentation.

This chapter includes synopses of the configuration variables that affect the master host. For more information about a variable, see *Control Center configuration variables* on page 75.

Control Center maintenance scripts on the master host

The scripts in the following list are installed when Control Center is installed, and are started either daily or weekly by `anacron`.

`/etc/cron.hourly/serviced`

This script invokes `logrotate` hourly, to manage the files in `/var/log/serviced`.

This script is required on the master host only.

`/etc/cron.daily/serviced`

This script invokes `logrotate` daily, to manage the `/var/log/serviced.access.log` file.

This script is required on the master host and on all delegate hosts.

`/etc/cron.weekly/serviced-fstrim`

This script invokes `fstrim` weekly, to reclaim unused blocks in the application data thin pool.

The life span of a solid-state drive (SSD) degrades when `fstrim` is run too frequently. If the block storage of the application data thin pool is an SSD, you can reduce the frequency at which this script is invoked, as long as the thin pool never runs out of free space. An identical copy of this script is located in `/opt/serviced/bin`.

This script is required on the master host only.

`/etc/cron.d/cron_zenossdbpack`

This script invokes `/opt/serviced/bin/serviced-zenossdbpack`, the database maintenance script for a Zenoss application, every Sunday at midnight. If the Zenoss application is not installed or is offline, the command fails. You can change the day of the week and time of day when the maintenance script is invoked by editing `/etc/cron.d/cron_zenossdbpack`.

This script is required on the master host only.

User access control

Control Center provides a browser interface and a command-line interface.

To gain access to the Control Center browser interface, users must have login accounts on the Control Center master host. In addition, users must be members of the Control Center browser interface access group, which by default is the system group, `wheel`. To enhance security, you may change the browser interface access group from `wheel` to any other group.

To use the Control Center command-line interface (CLI) on a Control Center host, a user must have login account on the host, and the account must be a member of the `serviced` group. The `serviced` group is created when the Control Center RPM package is installed.

Note You can use two different groups to control access to the browser interface and the CLI. You can enable access to both interfaces for the same users by choosing the `serviced` group as the browser interface access group.

Pluggable Authentication Modules (PAM) has been tested and is recommended for enabling access to both the browser interface and the command-line interface. However, the PAM configuration must include the `sudo` service. Control Center relies on the host's `sudo` configuration, and if no configuration is present, PAM defaults to the configuration for `other`, which is typically too restrictive for Control Center users. For more information about configuring PAM, refer to your operating system documentation.

Adding users to the default browser interface access group

Use this procedure to add users to the default browser interface access group of Control Center, `wheel`.

Note Perform this procedure or the next procedure, but not both.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Add a user to the `wheel` group.

   Replace *User* with the name of a login account on the master host.

   ```
   usermod -aG wheel User
   ```

   Repeat the preceding command for each user to add.

Configuring a regular group as the Control Center browser interface access group

Use this procedure to change the default browser interface access group of Control Center from `wheel` to a non-system group.

The following Control Center variables are used in this procedure:

SERVICED_ADMIN_GROUP

Default: `wheel`

The name of the Linux group on the `serviced` master host whose members are authorized to use the `serviced` browser interface. You may replace the default group with a group that does not have superuser privileges.

SERVICED_ALLOW_ROOT_LOGIN

Default: `1` (true)

Determines whether the `root` user account on the `serviced` master host may be used to gain access to the `serviced` browser interface.

Note Perform this procedure or the previous procedure, but not both.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Create a variable for the group to designate as the administrative group.

   In this example, the group is `ccuser`. You may choose a different group, or choose the `serviced` group. (Choosing the serviced group allows all browser interface users to use the CLI.)

   ```
   myGROUP=ccuser
   ```
3. Create a new group, if necessary.

   ```
   groupadd $myGROUP
   ```
4. Add one or more existing users to the group.

   Replace *User* with the name of a login account on the host:

   ```
   usermod -aG $myGROUP User
   ```

   Repeat the preceding command for each user to add.
5. Specify the new administrative group in the `serviced` configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_ADMIN_GROUP* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Change the value from `wheel` to the name of the group you chose earlier.
   e. Save the file, and then close the editor.
6. Optional: Prevent the `root` user from gaining access to the Control Center browser interface, if desired.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_ALLOW_ROOT_LOGIN* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Change the value from `1` to `0`.
   e. Save the file, and then close the editor.

Enabling use of the command-line interface

Use this procedure to enable a user to perform administrative tasks with the Control Center command-line interface.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Add a user to the `serviced` group.

   Replace *User* with the name of a login account on the host.

   ```
   usermod -aG serviced User
   ```

   Repeat the preceding command for each user to add.

Configuring the base size device for tenant data storage

Use this procedure to configure the base size of virtual storage devices for tenants in the application data thin pool. The base size is used each time a tenant device is created. In particular, the first time `serviced` starts, it creates the base size device and then creates a tenant device from the base size device.

The following Control Center configuration variable is used to specify the base size device:

SERVICED_DM_BASESIZE

Default: `100G`

The base size of virtual storage devices for tenants in the application data thin pool, in gigabytes. The units symbol (`G`) is required. This variable is used when `serviced` starts for the first time, to set the initial size of tenant devices, and when a backup is restored, to set the size of the restored tenant device.

The base size device is sparse device that occupies at most 1MB of space in the application data thin pool; its size has no immediate practical impact. However, the application data thin pool should have enough space for twice the size of each tenant device it supports, to store both the data itself and snapshots of the data. Since the application data thin pool is an LVM logical volume, its size can be increased at any time. Likewise, the size of a tenant device can be increased, as long as the available space in the thin pool can support the larger tenant device plus snapshots.

Perform these steps:

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Identify the size of the thin pool for application data.

   The size is required to set an accurate value for the *SERVICED_DM_BASESIZE* variable.

   ```
   lvs --options=lv_name,lv_size | grep serviced-pool
   ```
3. Edit storage variables in the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_DM_BASESIZE* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Change the value, if necessary.
      Replace *Fifty-Percent* with the value that is less than or equal to 50% of the size of the thin pool for application data. Include the symbol for gigabytes, `G`:

      ```
      SERVICED_DM_BASESIZE=Fifty-PercentG
      ```
   e. Save the file, and then close the editor.
4. Verify the settings in the `serviced` configuration file.

   ```
   grep -E '^\b*[A-Z_]+' /etc/default/serviced
   ```

Setting the host role to master

Use this procedure to configure a host as the master host. The following configuration variable assigns the host role:

SERVICED_MASTER

Default: `1` (true)

Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Only one `serviced` instance can be the master; all other instances must be delegates. The default value assigns the master role. To assign the delegate role, set the value to `0` (false). This variable must be explicitly set on all Control Center hosts.

Perform these steps:

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_MASTER* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Save the file, and then close the editor.
3. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Optional: Changing the local Docker registry endpoint

Use this procedure to configure the master host with the endpoint of an alternative local Docker registry. Control Center includes a local Docker registry, but you may use an existing registry in your environment, if desired. For more information about configuring a local Docker registry, please refer to *Docker documentation*.

Note Changing the local Docker registry endpoint is rare. Perform this procedure only if you are sure it is necessary and the alternative local Docker registry is already available in your environment.

The following configuration variable identifies the local Docker registry endpoint:

SERVICED_DOCKER_REGISTRY

Default: `localhost:5000`

The endpoint of the local Docker registry, which `serviced` uses to store internal services and application images.

If the default value is changed, the host's Docker configuration file must include the `--insecure-registry` flag with the same value as this variable.

The safest replacement for `localhost` is the IPv4 address of the registry host. Otherwise, the fully-qualified domain name of the host must be specified.

Perform these steps:

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_DOCKER_REGISTRY* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Replace `localhost:5000` with the endpoint of the local Docker registry.
      Use the IP address or fully-qualified domain name of the host and the port number.
   e. Save the file, and then close the editor.

3 Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

4 Add the insecure registry flag to the Docker configuration file.

a Open `/etc/sysconfig/docker` in a text editor.

b Add the following flag to the end of the *OPTIONS* declaration.

Replace *Registry-Endpoint* with the same value used for the *SERVICED_DOCKER_REGISTRY* variable:

```
--insecure-registry=Registry-Endpoint
```

Note Use a space character ( ) to separate flags, and make sure the double quote character (") delimits the declaration of *OPTIONS*.

c Save the file, and then close the editor.

5 Restart the Docker service.

```
systemctl restart docker
```

Optional: Configuring offline use

Use this procedure to configure a host to operate without internet access. The following configuration variable informs `serviced` that internet access is not available:

SERVICED_OUTBOUND_IP

Default: (none)

The IPv4 address that delegates use to connect to the master host. When no address is specified, `serviced` attempts to discover its public IP address by pinging `google.com`.

This variable must be set on all Control Center hosts in either of the following scenarios:

- Control Center is deployed behind a firewall and `google.com` is not reachable. Set the value to the IPv4 address of the master host.
- Control Center is deployed in a high-availability cluster. Set the value to the virtual IPv4 address of the high-availability cluster (*HA-Virtual-IP*).

Note Setting the Docker *HTTP_PROXY* or *HTTPS_PROXY* environment variables prevents access to the IP address defined with this variable. To enable access, unset the Docker variables, and then reboot the host.

Perform these steps:

1 Log in to the host as `root`, or as a user with superuser privileges.

2 Identify the IPv4 address of the host.

```
hostname -i
```

3 Edit the Control Center configuration file.

a Open `/etc/default/serviced` in a text editor.

b Locate the line for the *SERVICED_OUTBOUND_IP* variable, and then make a copy of the line, immediately below the original.

c Remove the number sign character (`#`) from the beginning of the line.

d Change the value to the IPv4 address identified in the previous step.
e Save the file, and then close the editor.

4 Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Master host configuration variables

The tables in this section provide an overview of the `serviced` configuration variables that apply to the Control Center master host. Set these variables as required for your environment or applications.

Best practices for configuration files

The Control Center configuration file, `/etc/default/serviced`, contains Bash environment variables that are read by the `serviced` daemon startup script. The following list describes recommended best practices for its use and maintenance:

1. When in doubt, make a backup. Before editing, making a backup copy of the configuration file is always the safest choice.
2. Copy a variable, then edit the copy. If you need to revert a variable to its default value, you don't have to leave the file to look it up.
3. Copy and edit a variable only if the default value needs to be changed. It's easier to troubleshoot problems when only non-default variables are copied and edited.
4. Put the first character of the variable declaration in the first column of its line. It's easier to `grep` for settings when each one starts a line.
5. Add customizations to the top of the file. Customizations at the end of the file or scattered throughout the file may be overlooked.
6. In high-availability deployments, the contents of `/etc/default/serviced` on the master nodes must be identical. Use a utility like `sum` to compare the files quickly.

Storage variables

The variables in the following table are set only on the master host.

- Use one of the first two groups of variables but not both.
- Before starting the master host for the first time, you might need to change the defaults of the third group.
- Typically, the defaults of the last two groups of variables are not changed until Control Center has managed an application for a while and a need arises.

The *SERVICED_STORAGE_STATS_UPDATE_INTERVAL* variable sets the interval for collecting kernel statistics about the application data thin pool. Its default value is unlikely to require a change until a need arises.

Variable	Description	Purpose
SERVICED_FS_TYPE *SERVICED_DM_ARGS* *SERVICED_DM_BASESIZE* *SERVICED_DM_THINPOOLDEV*	The specifications of a `devicemapper`-based application data storage resource for production use.	Provide basic information about the data storage resource.
SERVICED_FS_TYPE *SERVICED_DM_LOOPDATASIZE* *SERVICED_DM_LOOPMETADATASIZE*	The specifications of a `devicemapper`-based	Provide basic information about the data storage resource.

Variable	Description	Purpose
SERVICED_ALLOW_LOOP_BACK	application data storage resource for development use.	
SERVICED_ISVCS_PATH *SERVICED_VOLUMES_PATH* *SERVICED_BACKUPS_PATH*	The data storage paths of separate functional components of Control Center internal services.	Enable separate storage areas for one or more components. The default installation process puts all three components on the same device.
SERVICED_SNAPSHOT_TTL *SERVICED_SNAPSHOT_USE_PERCENT* *SERVICED_MAX_DFS_TIMEOUT*	The snapshot retention interval, the percentage of the data storage thin pool that is unused, and the snapshot attempt timeout interval.	Prevent the creation of snapshots that are too large to fit the thin pool.
SERVICED_LOGSTASH_MAX_DAYS *SERVICED_LOGSTASH_MAX_SIZE* *SERVICED_LOGSTASH_CYCLE_TIME*	The variables that manage the amount of space used by the application log storage service.	Prevent application logs from filling the storage device that logstash uses.

Internal services endpoint variables

The variables in the following table must be set identically on all Control Center delegate hosts.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Endpoint	Description
SERVICED_DOCKER_REGISTRY	(varies)	The local Docker registry for Control Center internal services images and application images.
SERVICED_ENDPOINT	*Master-Host*`:4979`	The `serviced` RPC server. The endpoint port number must match the value of *SERVICED_RPC_PORT*.
SERVICED_LOG_ADDRESS	*Master-Host*`:5042`	The *logstash* service.
SERVICED_LOGSTASH_ES	*Master-Host*`:9100`	The Elasticsearch service for logstash.
SERVICED_STATS_PORT	*Master-Host*`:8443`	The `serviced` metrics consumer service.
SERVICED_AUTH_TOKEN_EXPIRATION	(none)	The length of time a delegate authentication token is valid.

RPC service variables

The variables in the following table must be set identically on all Control Center hosts, except:

- *SERVICED_RPC_PORT*, set only on the master
- *SERVICED_MAX_RPC_CLIENTS*, set only on delegates

By default, `serviced` uses TLS to encrypt all RPC traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Where to set
SERVICED_ENDPOINT	Master, delegates
SERVICED_MAX_RPC_CLIENTS	Delegates
SERVICED_RPC_PORT	Master
SERVICED_RPC_CERT_VERIFY	Master, delegates
SERVICED_RPC_DISABLE_TLS	Master, delegates
SERVICED_RPC_TLS_MIN_VERSION	Master, delegates
SERVICED_RPC_TLS_CIPHERS	Master, delegates
SERVICED_KEY_FILE	Master
SERVICED_CERT_FILE	Master
SERVICED_RPC_DIAL_TIMEOUT	Master, delegates
SERVICED_AUTH_TOKEN_EXPIRATION	Master

Multiplexer variables

The variables in the following table must be set identically on all Control Center hosts.

By default, `serviced` uses TLS to encrypt all mux traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Where to set
SERVICED_MUX_PORT	Master, delegates
SERVICED_MUX_DISABLE_TLS	Master, delegates
SERVICED_MUX_TLS_MIN_VERSION	Master, delegates
SERVICED_MUX_TLS_CIPHERS	Master, delegates
SERVICED_KEY_FILE	Master
SERVICED_CERT_FILE	Master
SERVICED_AUTH_TOKEN_EXPIRATION	Master

HTTP server variables

The variables in the following table are set only on the master host, except the *SERVICED_UI_PORT* variable, which must be set identically on all Control Center hosts.

By default, `serviced` uses TLS to encrypt all HTTP traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

Variable	Description
SERVICED_UI_PORT	The port on which the HTTP server listens for requests.
SERVICED_TLS_MIN_VERSION	The minimum version of TLS that `serviced` accepts for HTTP traffic.
SERVICED_TLS_CIPHERS	The list TLS ciphers that `serviced` accepts for HTTP traffic.
SERVICED_KEY_FILE	The path of a digital certificate key file.
SERVICED_CERT_FILE	The path of a digital certificate file.

Browser interface variables (master host only)

The variables in the following table are set only on the master host.

Variable	Description
SERVICED_UI_POLL_FREQUENCY	The number of seconds between polls from browser interface clients.
SERVICED_SVCSTATS_CACHE_TIMEOUT	The number of seconds to cache statistics about services.
SERVICED_ADMIN_GROUP	The group on the master host whose members can use the browser interface.
SERVICED_ALLOW_ROOT_LOGIN	Determines whether `root` on the master host can use the browser interface.

Tuning variables (master host only)

Variable	Description
SERVICED_ES_STARTUP_TIMEOUT	The number of seconds to wait for the Elasticsearch service to start.
SERVICED_MASTER_POOLID	The name of the default resource pool. This variable is only used the first time `serviced` is started.

Universal configuration variables

The tables in this section provide an overview of the `serviced` configuration variables that apply to all Control Center hosts. Set these variables as required for your environment or applications.

Role variable

Variable	Description
SERVICED_MASTER	Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler

Variable	Description
	and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Browser interface variable (all hosts)

Variable	Description
SERVICED_UI_PORT	The port on which the HTTP server listens for requests.

Networking variables

Variable	Description
SERVICED_STATIC_IPS	A list of one or more static IP addresses for IP assignment.
SERVICED_OUTBOUND_IP	The IP address of the network interface for `serviced` to use. When this variable is not set, `serviced` uses the IP address of the default network interface and assumes it has internet access. To prevent `serviced` from assuming it has internet access, set this variable.
SERVICED_VIRTUAL_ADDRESS_SUBNET	The private network for containers that use virtual IP addresses. The default is `10.3.0.0/16`, and the network can be unique on each host. A `/29` network is sufficient.
SERVICED_DOCKER_DNS	A list of one or more DNS servers. The list is injected into all Docker containers.

Debugging variables

Variable	Description
SERVICED_LOG_LEVEL	The log level `serviced` uses when writing to the system log.
SERVICED_DEBUG_PORT	The port on which `serviced` listens for HTTP requests for the *Go profiler*.
SERVICED_DOCKER_LOG_DRIVER	The log driver for all Docker container logs.
SERVICED_DOCKER_LOG_CONFIG	Docker `--log-opt` options.

Tuning variables (all Control Center hosts)

Variable	Description
GOMAXPROCS	The maximum number of CPU cores that `serviced` uses.
SERVICED_MAX_CONTAINER_AGE	The number of seconds `serviced` waits before removing a stopped container.
SERVICED_ISVCS_ENV_[0-9]+	Startup arguments to pass to specific internal services.

Variable	Description
SERVICED_SERVICE_MIGRATION_TAG	Overrides the default value for the service migration image.
SERVICED_OPTS	Startup arguments for `serviced`.
SERVICED_CONTROLLER_BINARY	The path of the `serviced-controller` binary.
SERVICED_HOME	The path of the home directory for `serviced`.
SERVICED_ETC_PATH	The path of the directory for `serviced` configuration files.
SERVICED_VHOST_ALIASES	A list of hostname aliases for a host; for example, `localhost`.
SERVICED_ZK_CONNECT_TIMEOUT	The number of seconds Control Center waits for a connection to the lead ZooKeeper host.
SERVICED_ZK_PER_HOST_CONNECT_DELAY	The number of seconds Control Center waits before attempting to connect to the next host in its round-robin list of ZooKeeper hosts.
SERVICED_ZK_RECONNECT_START_DELAY, *SERVICED_ZK_RECONNECT_MAX_DELAY*	These are used together when Control Center is unable to re-establish a connection with the lead ZooKeeper host.

Starting Control Center for the first time

Use this procedure to start the Control Center service (`serviced`) on a master host after installing and configuring it. This procedure is valid for single-host and multi-host deployments, whether the deployment has internet access or not, and is only performed once.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

3. Start the Control Center service.

```
systemctl start serviced
```

To monitor progress, enter the following command:

```
journalctl -flu serviced -o cat
```

The `serviced` daemon tags images in the local Docker registry and starts its internal services. The Control Center browser and command-line interfaces are unavailable for about 3 minutes.

When the message `Host Master successfully started` is displayed, Control Center is ready for the next procedure.

Note Until the master host is added to a pool, all `serviced` CLI commands that use the RPC server must be run as `root`.

Adding the master host to a resource pool

Perform one of the procedures in this section.

Adding the master host to a resource pool (single-host deployments)

Use this procedure to add the master host to a resource pool. For single-host deployments, the master host is added to the `default` pool.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Add the master host to the `default` resource pool.
   Replace *Hostname-Or-IP* with the hostname or IP address of the Control Center master host:

   ```
   serviced host add --register Hostname-Or-IP:4979 default
   ```

   If you use a hostname, all Control Center hosts must be able to resolve the name, either through an entry in `/etc/hosts` or through a nameserver on the network.

Adding the master host to a resource pool (multi-host deployments)

Use this procedure to add the master host to a resource pool. For multi-host deployments, the master host is added to its own, separate pool.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Create a new resource pool named `master`.

   ```
   serviced pool add master
   ```

3. Assign administrative and distributed file system (DFS) permissions to the new resource pool.

   ```
   serviced pool set-permission --admin --dfs master
   ```

4. Add the master host to the new resource pool.
   Replace *Hostname-Or-IP* with the hostname or IP address of the Control Center master host:

   ```
   serviced host add --register Hostname-Or-IP:4979 master
   ```

   If you use a hostname, all Control Center hosts must be able to resolve the name, either through an entry in `/etc/hosts` or through a nameserver on the network.

4 Installing delegate hosts

This chapter describes how to install Control Center on a Red Hat Enterprise Linux (RHEL) or CentOS host. The candidate host must have the CPU, RAM, and storage resources required to serve as a Control Center delegate host.

For more information about delegate host requirements, refer to the *Control Center Planning Guide*.

Note If you are installing a single-host deployment, skip this chapter.

Repeat the procedures in this chapter on each host to add to your deployment.

Verifying candidate host resources

Use this procedure to determine whether a host's hardware resources and operating system are sufficient to serve as a Control Center delegate host. Perform this procedure on each candidate delegate host in your deployment.

1. Log in to the candidate host as `root`, or as a user with superuser privileges.
2. Verify that the host implements the 64-bit version of the x86 instruction set.

   ```
   uname -m
   ```

   - If the output is `x86_64`, the architecture is 64-bit. Proceed to the next step
   - If the output is `i386/i486/i586/i686`, the architecture is 32-bit. Stop this procedure and select a different host.
3. Verify that the host has adequate storage space on the root filesystem.

   ```
   df -h /
   ```

   If the result does not include a minimum of 35GB of available space, stop this procedure and select a different host.
4. Determine whether the available memory and swap is sufficient.
   a. Display the available memory.

      ```
      free -h
      ```

   b. Compare the available memory with the amount required for a delegate host in your deployment.

      For more information, refer to the *Control Center Planning Guide*.

   If the result does not meet minimum requirements, stop this procedure and select a different host.

5 Determine whether the installed operating system release is one of the releases that has been tested with Control Center.

```
cat /etc/redhat-release
```

- If the result includes `7.2`, `7.3`, or `7.4` proceed to the next step.
- If the result does not include `7.2`, `7.3`, or `7.4`, select a different host, and then start this procedure again.

6 Determine whether the CPU resources are sufficient.

a Display the total number of CPU cores.

```
cat /proc/cpuinfo | grep -Ec '^core id'
```

b Compare the available resources with the requirements for a Control Center master host.

For more information, refer to the *Control Center Planning Guide*.

7 Determine whether the CPU resources support the AES instruction set.

```
cat /proc/cpuinfo | grep -Ec '^flags.*aes'
```

For optimal performance, the result of the preceding commands must match the total number of CPU resources available on the host. If the result is 0, performance is severely degraded.

If the result is 0 and the candidate host is a virtual machine, the managing hypervisor may be configured in Hyper-V compatibility mode. Check the setting and disable it, if possible, or select a different host.

8 Ensure the host has a persistent numeric ID.

Each Control Center host must have a unique host ID, and the ID must be persistent (not change when the host reboots).

```
test -f /etc/hostid || genhostid ; hostid
```

Record the ID for comparison with other Control Center hosts.

9 Verify that name resolution works on this host.

```
hostname -i
```

If the result is not a valid IPv4 address, add an entry for the host to the network nameserver, or to `/etc/hosts`.

10 Add an entry to `/etc/hosts` for localhost, if necessary.

a Determine whether `127.0.0.1` is mapped to `localhost`.

```
grep 127.0.0.1 /etc/hosts | grep localhost
```

If the preceding commands return no result, perform the following substep.

b Add an entry to `/etc/hosts` for `localhost`.

```
echo "127.0.0.1 localhost" >> /etc/hosts
```

11 Update the Linux kernel, if necessary.

a Determine which kernel version is installed.

```
uname -r
```

If the result is lower than `3.10.0-327.22.2.el7.x86_64`, perform the following substep.

b Update the kernel, and then restart the host.

The following commands require internet access or a local mirror of operating system packages.

```
yum makecache fast && yum update kernel && reboot
```

Delegate host storage requirements

This section provides a quick reference to the block storage requirements of Control Center delegate hosts. For more information, refer to the *Control Center Planning Guide*.

Control Center hosts need either unformatted block storage devices or partitions, or free space in one or more LVM volume groups.

- Enter the following command to display information about block storage:

```
lsblk -ap --output=NAME,SIZE,TYPE,FSTYPE,MOUNTPOINT
```

- Enter the following command to display information about LVM volume groups:

```
vgdisplay
```

The following storage configuration is recommended for delegate hosts:

- A root filesystem with a minimum of 35GB of storage, formatted with XFS.
- A dedicated swap area.
- One or more block devices or partitions, or one or more LVM physical volumes, with a total of 50GB of space. A step in this chapter configures the space as an LVM thin pool for Docker data.

Preparing a delegate host

Perform the steps in *Downloading and staging required files* on page 8, before performing this procedure.

Use this procedure to prepare a RHEL/CentOS host as a Control Center delegate host.

1. Log in to the candidate delegate host as `root`, or as a user with superuser privileges.
2. Disable the firewall, if necessary.

   This step is required for installation but not for deployment. For more information, refer to the *Control Center Planning Guide*.

   a Determine whether the `firewalld` service is enabled.

   ```
   systemctl status firewalld.service
   ```

   - If the result includes `Active: inactive (dead)`, the service is disabled. Proceed to the next step.
   - If the result includes `Active: active (running)`, the service is enabled. Perform the following substep.

   b Disable the `firewalld` service.

   ```
   systemctl stop firewalld && systemctl disable firewalld
   ```

On success, the preceding commands display messages similar to the following example:

```
rm '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
rm '/etc/systemd/system/basic.target.wants/firewalld.service'
```

3 Optional: Enable persistent storage for log files, if desired.

By default, RHEL/CentOS systems store log data only in memory or in a ring buffer in the `/run/log/journal` directory. By performing this step, log data persists and can be saved indefinitely, if you implement log file rotation practices. For more information, refer to your operating system documentation.

Note The following commands are safe when performed during an installation, before Docker or Control Center are installed or running. To enable persistent log files after installation, stop Control Center, stop Docker, and then enter the following commands.

```
mkdir -p /var/log/journal && systemctl restart systemd-journald
```

4 Enable and start the Dnsmasq package.

The package facilitates networking among Docker containers.

```
systemctl enable dnsmasq && systemctl start dnsmasq
```

If name resolution in your environment relies solely on entries in `/etc/hosts`, configure `dsnmasq` so that containers can use the file:

a Open `/etc/dnsmasq.conf` with a text editor.

b Locate the line that starts with `#domain-needed`, and then make a copy of the line, immediately below the original.

c Remove the number sign character (`#`) from the beginning of the line.

d Locate the line that starts with `#bogus-priv`, and then make a copy of the line, immediately below the original.

e Remove the number sign character (`#`) from the beginning of the line.

f Locate the line that starts with `#local=/localnet/`, and then make a copy of the line, immediately below the original.

g Remove `net`, and then remove the number sign character (`#`) from the beginning of the line.

h Locate the line that starts with `#domain=example.com`, and then make a copy of the line, immediately below the original.

i Replace `example.com` with `local`, and then remove the number sign character (`#`) from the beginning of the line.

j Save the file, and then close the editor.

k Restart the `dnsmasq` service.

```
systemctl restart dnsmasq
```

5 Install and configure the NTP package.

Note This procedure assumes the host has internet access. To install and configure NTP on a host that does not have internet access, see *Configuring a private master NTP server* on page 86.

a Install the package.

```
yum install ntp
```

b Set the system time.

```
ntpd -gq
```

c Enable the `ntpd` daemon.

```
systemctl enable ntpd
```

d Configure `ntpd` to start when the system starts.

Currently, an unresolved issue associated with NTP prevents `ntpd` from restarting correctly after a reboot. The following commands provide a workaround to ensure that it does.

```
echo "systemctl start ntpd" >> /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
```

Installing Docker CE and Control Center

Perform the steps in *Downloading and staging required files* on page 8 before performing this procedure.

Use this procedure to install Docker CE and Control Center on a host.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Install Docker CE 17.09.0 from the local repository mirror.
   a Install Docker CE.

   ```
   yum install --enablerepo=zenoss-mirror docker-ce-17.09.0.ce
   ```

   If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 89 for potential resolutions.
   b Enable automatic startup.

   ```
   systemctl enable docker
   ```
3. Install Control Center 1.5.0 from the local repository mirror.
   a Install Control Center.

   ```
   yum install --enablerepo=zenoss-mirror \
     /opt/zenoss-repo-mirror/serviced-1.5.0-1.x86_64.rpm
   ```

   If `yum` returns an error due to dependency issues, see *Resolving package dependency conflicts* on page 89 for potential resolutions.
   b Enable automatic startup.

   ```
   systemctl enable serviced
   ```
4. Make a backup copy of the Control Center configuration file.
   a Make a copy of `/etc/default/serviced`.

   ```
   cp /etc/default/serviced /etc/default/serviced-1.5.0-orig
   ```

   b Set the backup file permissions to read-only.

   ```
   chmod 0440 /etc/default/serviced-1.5.0-orig
   ```

Configuring NFS 4.0

Use this procedure to configure NFS 4.0 on delegate hosts if the operating system release is 7.4. There may be a file locking defect in NFS 4.1 with RHEL/CentOS 7.4.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Determine which release is installed.

```
cat /etc/redhat-release
```

   - If the result includes `7.4`, perform the remaining steps of this procedure.
   - If the result includes `7.2` or `7.3`, continue to the next procedure.
3. Change the NFS configuration file.
   a. Open `/etc/nfsmount.conf` with a text editor.
   b. Locate the `Defaultvers` directive.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Change the value from `4` to `4.0`.
   The line should appear as follows:

```
Defaultvers=4.0
```

   e. Save the file, and then close the editor.
4. Restart the NFS server.

```
systemctl restart nfs-server
```

Configuring Docker

Use this procedure to configure Docker.

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Create a symbolic link for the Docker temporary directory.

   Docker uses its temporary directory to spool images. The default directory is `/var/lib/docker/tmp`. The following command specifies the same directory that Control Center uses, `/tmp`. You can specify any directory that has a minimum of 10GB of unused space.
   a. Create the `docker` directory in `/var/lib`.

```
mkdir /var/lib/docker
```

   b. Create the link to `/tmp`.

```
ln -s /tmp /var/lib/docker/tmp
```

3. Create a `systemd` drop-in file for Docker.
   a. Create the override directory.

```
mkdir -p /etc/systemd/system/docker.service.d
```

   b. Create the unit drop-in file.

```
cat <<EOF > /etc/systemd/system/docker.service.d/docker.conf
```

```
[Service]
TimeoutSec=300
EnvironmentFile=-/etc/sysconfig/docker
ExecStart=
ExecStart=/usr/bin/dockerd \$OPTIONS
TasksMax=infinity
EOF
```

c Reload the `systemd` manager configuration.

```
systemctl daemon-reload
```

4 Create an LVM thin pool for Docker data.

For more information about the `serviced-storage` command, see *serviced-storage* on page 71.

To use an entire block device or partition for the thin pool, replace *Device-Path* with the device path:

```
serviced-storage create-thin-pool docker Device-Path
```

To use 50GB of an LVM volume group for the thin pool, replace *Volume-Group* with the name of an LVM volume group:

```
serviced-storage create-thin-pool --size=50G docker Volume-Group
```

On success, the result is the device mapper name of the thin pool, which always starts with `/dev/mapper`.

5 Configure and start the Docker service.

a Create a variable for the name of the Docker thin pool.

Replace *Thin-Pool-Device* with the name of the thin pool device created in the previous step:

```
myPool="Thin-Pool-Device"
```

b Create variables for adding arguments to the Docker configuration file. The `--exec-opt` argument is a workaround for *a Docker issue* on RHEL/CentOS 7.x systems.

```
myDriver="--storage-driver devicemapper"
myLog="--log-level=error"
myFix="--exec-opt native.cgroupdriver=cgroupfs"
myMount="--storage-opt dm.mountopt=discard"
myFlag="--storage-opt dm.thinpooldev=$myPool"
```

c Add the arguments to the Docker configuration file.

```
echo 'OPTIONS="'$myLog $myDriver $myFix $myMount $myFlag'"' \
  >> /etc/sysconfig/docker
```

d Start or restart Docker.

```
systemctl restart docker
```

The startup may take up to a minute, and may fail. If startup fails, repeat the `restart` command.

6 Configure name resolution in containers.

Each time it starts, `docker` selects an IPv4 subnet for its virtual Ethernet bridge. The selection can change; this step ensures consistency.

a Identify the IPv4 subnet and netmask `docker` has selected for its virtual Ethernet bridge.

```
ip addr show docker0 | grep inet
```

b Open `/etc/sysconfig/docker` in a text editor.

c Add the following flags to the end of the *OPTIONS* declaration.

Replace *Bridge-Subnet* with the IPv4 subnet `docker` selected for its virtual bridge:

```
--dns=Bridge-Subnet --bip=Bridge-Subnet/16
```

For example, if the bridge subnet is 172.17.0.1, add the following flags:

```
--dns=172.17.0.1 --bip=172.17.0.1/16
```

Note Use a space character ( ) to separate flags, and make sure the double quote character (") delimits the declaration of *OPTIONS*.

d Save the file, and then close the editor.

e Restart the Docker service.

```
systemctl restart docker
```

Proceed to the next chapter and configure the host.

Configuring and starting delegate hosts 5

This chapter provides the procedures that are required to configure a delegate host, describes the configuration options that apply to delegate hosts, and includes steps for starting a delegate for the first time. Perform the procedures in this chapter on each Control Center delegate host.

Note If you are installing a single-host deployment, skip this chapter.

This chapter includes synopses of the configuration variables that affect delegate hosts. For more information about a variable, see *Control Center configuration variables* on page 75.

Control Center maintenance scripts on delegate hosts

The scripts in the following list are installed when Control Center is installed, and are started either daily or weekly by `anacron`.

Of these scripts, only the first is required on delegate hosts. The others should be removed.

`/etc/cron.hourly/serviced`

This script invokes `logrotate` hourly, to manage the `/var/log/serviced-audit.log` and `/var/log/application-audit.log` files.

This script should be removed from delegate hosts.

`/etc/cron.daily/serviced`

This script invokes `logrotate` daily, to manage the `/var/log/serviced.access.log` file.

This script is required on the master host and on all delegate hosts.

`/etc/cron.weekly/serviced-fstrim`

This script should be removed from delegate hosts.

`/etc/cron.d/cron_zenossdbpack`

This script should be removed from delegate hosts.

Enabling use of the command-line interface

Use this procedure to enable a user to perform administrative tasks with the Control Center command-line interface.

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Add a user to the `serviced` group.

Replace *User* with the name of a login account on the host.

```
usermod -aG serviced User
```

Repeat the preceding command for each user to add.

Setting the host role to delegate

Use this procedure to configure a host as a delegate host. The following configuration variable assigns the host role:

SERVICED_MASTER

Default: `1` (true)

Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Only one `serviced` instance can be the master; all other instances must be delegates. The default value assigns the master role. To assign the delegate role, set the value to `0` (false). This variable must be explicitly set on all Control Center hosts.

Perform these steps:

1. Log in to the host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_MASTER* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Change the value from `1` to `0`.
   e. Save the file, and then close the editor.
3. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Changing the local Docker registry endpoint

Use this procedure to configure the delegate host with the endpoint of the local Docker registry. Unless the master host is configured with an alternative local Docker registry, which is rare, the endpoint is the master host's hostname or IP address and port 5000.

The following configuration variable identifies the local Docker registry endpoint:

SERVICED_DOCKER_REGISTRY

Default: `localhost:5000`

The endpoint of the local Docker registry, which `serviced` uses to store internal services and application images.

If the default value is changed, the host's Docker configuration file must include the `--insecure-registry` flag with the same value as this variable.

The safest replacement for `localhost` is the IPv4 address of the registry host. Otherwise, the fully-qualified domain name of the host must be specified.

Perform these steps:

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Locate the line for the *SERVICED_DOCKER_REGISTRY* variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Replace `localhost:5000` with the endpoint of the local Docker registry.
      If the master host is configured with an alternative local Docker registry, use the same endpoint here. Otherwise, just replace `localhost` with the IP address or fully-qualified domain name of the Control Center master host.
   e. Save the file, and then close the editor.
3. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

4. Add the insecure registry flag to the Docker configuration file.
   a. Open `/etc/sysconfig/docker` in a text editor.
   b. Add the local Docker registry endpoint to the end of the *OPTIONS* declaration.
      Replace *Registry-Endpoint* with the same value used for the *SERVICED_DOCKER_REGISTRY* variable:

```
--insecure-registry=Registry-Endpoint
```

      Note Use a space character ( ) to separate flags, and make sure the double quote character (`"`) delimits the declaration of *OPTIONS*.
   c. Save the file, and then close the editor.
5. Restart the Docker service.

```
systemctl restart docker
```

Setting internal services endpoints

Use this procedure to configure a delegate host with the endpoints of the Control Center internal services.

The following configuration variables identify the internal services endpoints:

SERVICED_ZK

Default: (none)

The list of endpoints in the `serviced` ZooKeeper ensemble, separated by the comma character (`,`). Each endpoint identifies an ensemble node. Each Control Center server and in-container proxy uses *SERVICED_ZK* to create a randomized, round-robin list, and cycles through the list when it attempts to establish a connection with the lead ZooKeeper host.

SERVICED_ENDPOINT

Default: `{{SERVICED_MASTER_IP}}:4979`

The endpoint of the `serviced` RPC server. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host. The port number of this endpoint must match the value of the *SERVICED_RPC_PORT* variable defined on the `serviced` master host.

SERVICED_LOG_ADDRESS

Default: `{{SERVICED_MASTER_IP}}:5042`

The endpoint of the logstash service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

SERVICED_LOGSTASH_ES

Default: `{{SERVICED_MASTER_IP}}:9100`

The endpoint of the Elasticsearch service for logstash. On delegate hosts, replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the Elasticsearch host, which by default is the `serviced` master host.

SERVICED_STATS_PORT

Default: `{{SERVICED_MASTER_IP}}:8443`

The endpoint of the `serviced` metrics consumer service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

Perform these steps:

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Edit the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. For each endpoint variable, locate the line that sets the variable, and then make a copy of the line, immediately below the original.
   c. Remove the number sign character (`#`) from the beginning of the line.
   d. Replace `localhost` or `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the master host.
   e. Save the file, and then close the editor.
3. Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Optional: Configuring offline use

Use this procedure to configure a host to operate without internet access. The following configuration variable informs `serviced` that internet access is not available:

SERVICED_OUTBOUND_IP

Default: (none)

The IPv4 address that delegates use to connect to the master host. When no address is specified, `serviced` attempts to discover its public IP address by pinging `google.com`.

This variable must be set on all Control Center hosts in either of the following scenarios:

- Control Center is deployed behind a firewall and `google.com` is not reachable. Set the value to the IPv4 address of the master host.
- Control Center is deployed in a high-availability cluster. Set the value to the virtual IPv4 address of the high-availability cluster (*HA-Virtual-IP*).

Note Setting the Docker *HTTP_PROXY* or *HTTPS_PROXY* environment variables prevents access to the IP address defined with this variable. To enable access, unset the Docker variables, and then reboot the host.

Perform these steps:

1. Log in to the host as `root`, or as a user with superuser privileges.

2 Identify the IPv4 address of the host.

```
hostname -i
```

3 Edit the Control Center configuration file.
   a Open `/etc/default/serviced` in a text editor.
   b Locate the line for the *SERVICED_OUTBOUND_IP* variable, and then make a copy of the line, immediately below the original.
   c Remove the number sign character (`#`) from the beginning of the line.
   d Change the value to the IPv4 address identified in the previous step.
   e Save the file, and then close the editor.
4 Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

Delegate host configuration variables

The tables in this section provide an overview of the `serviced` configuration variables that apply to Control Center delegate hosts. Set these variables as required for your environment or applications.

Delegate variables

The following miscellaneous variables apply only to delegate hosts.

Variable	Description
SERVICED_ZK	The list of hosts in the ZooKeeper ensemble.
SERVICED_STATS_PERIOD	The frequency at which delegates gather metrics to send to the master host.
SERVICED_IPTABLES_MAX_CONNECTIONS	The maximum number of open connections to allow on a delegate. The number increases when the master is unavailable and decreases when the master returns to service.

Internal services endpoint variables

The variables in the following table must be set identically on all Control Center delegate hosts.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Endpoint	Description
SERVICED_DOCKER_REGISTRY	(varies)	The local Docker registry for Control Center internal services images and application images.
SERVICED_ENDPOINT	*Master-Host*`:4979`	The `serviced` RPC server. The endpoint port number must match the value of *SERVICED_RPC_PORT*.
SERVICED_LOG_ADDRESS	*Master-Host*`:5042`	The *logstash* service.

Variable	Endpoint	Description
SERVICED_LOGSTASH_ES	*Master-Host*:9100	The Elasticsearch service for logstash.
SERVICED_STATS_PORT	*Master-Host*:8443	The `serviced` metrics consumer service.
SERVICED_AUTH_TOKEN_EXPIRATION	(none)	The length of time a delegate authentication token is valid.

RPC service variables

The variables in the following table must be set identically on all Control Center hosts, except:

- *SERVICED_RPC_PORT*, set only on the master
- *SERVICED_MAX_RPC_CLIENTS*, set only on delegates

By default, `serviced` uses TLS to encrypt all RPC traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Where to set
SERVICED_ENDPOINT	Master, delegates
SERVICED_MAX_RPC_CLIENTS	Delegates
SERVICED_RPC_PORT	Master
SERVICED_RPC_CERT_VERIFY	Master, delegates
SERVICED_RPC_DISABLE_TLS	Master, delegates
SERVICED_RPC_TLS_MIN_VERSION	Master, delegates
SERVICED_RPC_TLS_CIPHERS	Master, delegates
SERVICED_KEY_FILE	Master
SERVICED_CERT_FILE	Master
SERVICED_RPC_DIAL_TIMEOUT	Master, delegates
SERVICED_AUTH_TOKEN_EXPIRATION	Master

Multiplexer variables

The variables in the following table must be set identically on all Control Center hosts.

By default, `serviced` uses TLS to encrypt all mux traffic. The *SERVICED_KEY_FILE* and *SERVICED_CERT_FILE* variables identify the digital certificate used for RPC, mux, and HTTP traffic.

The *SERVICED_AUTH_TOKEN_EXPIRATION* variable affects RPC, mux, and internal services endpoint traffic.

Variable	Where to set
SERVICED_MUX_PORT	Master, delegates
SERVICED_MUX_DISABLE_TLS	Master, delegates
SERVICED_MUX_TLS_MIN_VERSION	Master, delegates
SERVICED_MUX_TLS_CIPHERS	Master, delegates
SERVICED_KEY_FILE	Master
SERVICED_CERT_FILE	Master
SERVICED_AUTH_TOKEN_EXPIRATION	Master

Universal configuration variables

The tables in this section provide an overview of the `serviced` configuration variables that apply to all Control Center hosts. Set these variables as required for your environment or applications.

Role variable

Variable	Description
SERVICED_MASTER	Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Browser interface variable (all hosts)

Variable	Description
SERVICED_UI_PORT	The port on which the HTTP server listens for requests.

Networking variables

Variable	Description
SERVICED_STATIC_IPS	A list of one or more static IP addresses for IP assignment.
SERVICED_OUTBOUND_IP	The IP address of the network interface for `serviced` to use. When this variable is not set, `serviced` uses the IP address of the default network interface and assumes it has internet access. To prevent `serviced` from assuming it has internet access, set this variable.
SERVICED_VIRTUAL_ADDRESS_SUBNET	The private network for containers that use virtual IP addresses. The default is `10.3.0.0/16`, and the network can be unique on each host. A `/29` network is sufficient.
SERVICED_DOCKER_DNS	A list of one or more DNS servers. The list is injected into all Docker containers.

Debugging variables

Variable	Description
SERVICED_LOG_LEVEL	The log level `serviced` uses when writing to the system log.
SERVICED_DEBUG_PORT	The port on which `serviced` listens for HTTP requests for the *Go profiler*.
SERVICED_DOCKER_LOG_DRIVER	The log driver for all Docker container logs.
SERVICED_DOCKER_LOG_CONFIG	Docker `--log-opt` options.

Tuning variables (all Control Center hosts)

Variable	Description
GOMAXPROCS	The maximum number of CPU cores that `serviced` uses.
SERVICED_MAX_CONTAINER_AGE	The number of seconds `serviced` waits before removing a stopped container.
SERVICED_ISVCS_ENV_[0-9]+	Startup arguments to pass to specific internal services.
SERVICED_SERVICE_MIGRATION_TAG	Overrides the default value for the service migration image.
SERVICED_OPTS	Startup arguments for `serviced`.
SERVICED_CONTROLLER_BINARY	The path of the `serviced-controller` binary.
SERVICED_HOME	The path of the home directory for `serviced`.
SERVICED_ETC_PATH	The path of the directory for `serviced` configuration files.
SERVICED_VHOST_ALIASES	A list of hostname aliases for a host; for example, `localhost`.
SERVICED_ZK_CONNECT_TIMEOUT	The number of seconds Control Center waits for a connection to the lead ZooKeeper host.
SERVICED_ZK_PER_HOST_CONNECT_DELAY	The number of seconds Control Center waits before attempting to connect to the next host in its round-robin list of ZooKeeper hosts.
SERVICED_ZK_RECONNECT_START_DELAY, *SERVICED_ZK_RECONNECT_MAX_DELAY*	These are used together when Control Center is unable to re-establish a connection with the lead ZooKeeper host.

Starting Control Center

Use this procedure to start `serviced` on a delegate host for the first time.

1 Log in to the delegate host as `root`, or as a user with superuser privileges.

2 Verify the settings in the `serviced` configuration file.

```
grep -E '^\b*[A-Z_]+' /etc/default/serviced
```

3 Start the Control Center service (`serviced`).

```
systemctl start serviced
```

To monitor progress, enter the following command:

```
journalctl -flu serviced -o cat
```

Delegate host authentication

Control Center uses RSA key pairs to create the authentication tokens that are required for all delegate communications. When you add a host to a resource pool, the `serviced` instance on the master host creates a private key for the delegate and bundles it with its own public key. The `serviced` instance on the delegate host uses the bundle to sign messages with its unique tokens.

Key bundles are installed by using an SSH connection or a file.

- The command to add a host to a pool can initiate an SSH connection with the delegate and install the key bundle. This option is the most secure, because no file is created. However, it requires either public key authentication or password authentication between the master and delegate hosts.
- When no SSH connection is requested, the command to add a host to a pool creates a file containing the key bundle. You can move the key bundle file to the delegate host with any file transfer method, and then install it on the delegate.

The following procedures demonstrate how to add a host to a resource pool and install its key bundle.

Adding a delegate host through an SSH connection

To succeed, the following statements about the login account used to perform this procedure must be true:

- The account exists on both the master host and on the delegate host.
- The account has `serviced` CLI privileges.
- The account has either public key authentication or password authentication enabled on the master host and on the delegate host.

Use this procedure to add a delegate host to a resource pool through an SSH connection.

1 Log in to the Control Center master host as a user with `serviced` CLI privileges.
2 Optional: Create a new resource pool:
   a Display the names of the existing resource pools.

   ```
   serviced pool list
   ```

   b Create a new resource pool.
   Replace *Resource-Pool* with the name of a new resource pool:

   ```
   serviced pool add Resource-Pool
   ```

   c Optional: Set permissions on the new resource pool, if desired.

Replace *Resource-Pool* with the name of the new resource pool:

```
serviced pool set-permission --dfs=true --admin=true Resource-Pool
```

3 Add a delegate host to a resource pool.

If the master and delegate host are configured for key-based access, the following command does not prompt you to add the delegate to the list of known hosts or to provide the password of the remote user account.

Use the hostname or IP address to identify a Control Center host. If you use a hostname, all Control Center hosts must be able to resolve it, either through an entry in `/etc/hosts` or through a nameserver on the network. In the following example, replace *Hostname-Or-IP* with the hostname or IP address of a delegate host, and replace *Resource-Pool* with the name of a resource pool.

If the host is behind a router or firewall for network address translation (NAT), include the option `--nat-address` to specify the NAT device's hostname or IP address and port of the delegate host.

```
serviced host add --register Hostname-Or-IP:4979 Resource-Pool \
    --nat-address==NAT-Hostname-Or-IP:NAT-Port
```

Adding a delegate host using a file

Use this procedure to add a delegate host to a resource pool by using a key bundle file.

1 Log in to the Control Center master host as a user with `serviced` CLI privileges.
2 Optional: Create a new resource pool, if desired.
   a Display the names of the existing resource pools.

```
serviced pool list
```

   b Create a new resource pool.
   Replace *Resource-Pool* with the name of a new resource pool:

```
serviced pool add Resource-Pool
```

   c Optional: Set permissions on the new resource pool, if desired.
   Replace *Resource-Pool* with the name of the new resource pool:

```
serviced pool set-permission --dfs=true --admin=true Resource-Pool
```

3 Add a delegate host to a resource pool.

Use the hostname or IP address to identify a Control Center host. If you use a hostname, all Control Center hosts must be able to resolve it, either through an entry in `/etc/hosts` or through a nameserver on the network. In the following example, replace *Hostname-Or-IP* with the hostname or IP address of a delegate host, and replace *Resource-Pool* with the name of a resource pool.

If the host is behind a router or firewall for network address translation (NAT), include the option `--nat-address` to specify the NAT device's hostname or IP address and port of the delegate host.

```
serviced host add Hostname-Or-IP:4979 Resource-Pool \
    --nat-address==NAT-Hostname-Or-IP:NAT-Port
```

The command creates a unique key bundle file in the local directory.

4 Use a file transfer utility such as `scp` to copy the key bundle file to the delegate host.

Once copied to the delegate host, the key bundle file is not needed on the master host and can be deleted.

5 Log in to the Control Center delegate host as a user with `serviced` CLI privileges.

6 Install the key bundle.
Replace *Key-Bundle-Path* with the pathname of the key bundle file:

```
serviced host register Key-Bundle-Path
```

7 Delete the key bundle file.
The file is no longer needed on the delegate host.
Replace *Key-Bundle-Path* with the pathname of the key bundle file:

```
rm Key-Bundle-Path
```

6 Configuring a ZooKeeper ensemble

This chapter describes how to create a ZooKeeper ensemble (cluster) for a multi-host Control Center deployment that includes a minimum of three hosts. If your deployment includes just one host or two hosts, skip this chapter.

ZooKeeper and Control Center

Control Center relies on *Apache ZooKeeper* to distribute and manage application services. ZooKeeper maintains the definitions of each service and the list of services assigned to each host. The scheduler, which runs on the master host, determines assignments and sends them to the ZooKeeper node that is serving as the ensemble leader. The leader replicates the assignments to the other ensemble nodes, so that the other nodes can assume the role of leader if the leader node fails.

All Control Center hosts retrieve assignments and service definitions from the ZooKeeper ensemble leader and then start services in Docker containers as required. So, the Control Center configuration files of all Control Center hosts must include a definition for the *SERVICED_ZK* variable, which specifies the ZooKeeper endpoints of the ensemble nodes. Additional variables are required on ensemble nodes.

A ZooKeeper ensemble requires a minimum of three nodes, which is sufficient for most environments. An odd number of nodes is recommended and an even number of nodes is strongly discouraged. A five-node ensemble improves failover protection during maintenance windows but larger ensembles yield no benefits.

The Control Center master host is always an ensemble node. All ensemble nodes should be on the same subnet.

Understanding the configuration process

The procedures in this chapter instruct you to create temporary variables that are used as building blocks, to construct Control Center configuration variables accurately. You append the Control Center variables to `/etc/default/serviced`, and then edit the file to move the variables to more appropriate locations.

The most important temporary variables specify the IP address or hostname of each host in the ZooKeeper ensemble. The following table identifies these important variables, the names and values of which must be identical on every Control Center host.

Variable name	Placeholder value	Actual value
`node1`	*Master*	The IP address or hostname of the master host.
`node2`	*Delegate-A*	The IP address or hostname of delegate host A.

Variable name	Placeholder value	Actual value
`node3`	*Delegate-B*	The IP address or hostname of delegate host B.

Note All ensemble hosts should be on the same subnet.

ZooKeeper variables

The variables in the following table are set only on ZooKeeper ensemble nodes, except *SERVICED_ZK*, which must be identical on all Control Center hosts.

Variable	Where to set
SERVICED_ISVCS_START	ZooKeeper ensemble nodes
SERVICED_ISVCS_ZOOKEEPER_ID	ZooKeeper ensemble nodes
SERVICED_ISVCS_ZOOKEEPER_QUORUM	ZooKeeper ensemble nodes
SERVICED_ZK	All Control Center hosts
SERVICED_ZK_SESSION_TIMEOUT	ZooKeeper ensemble nodes

Example multi-host ZooKeeper configuration

This example shows the ZooKeeper variables in the `/etc/default/serviced` configuration file of each host in a 4-node Control Center deployment. For convenience, the relevant settings for each node or host are also included in subsequent procedures.

Note The value of the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable is formatted to fit the available space. In the configuration file, the variable and value are on the same line.

Master host and ZooKeeper ensemble node, 198.51.100.135:

```
SERVICED_ISVCS_ZOOKEEPER_ID=1
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@0.0.0.0:2888:3888,\
  2@198.51.100.136:2888:3888,3@198.51.100.137:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Delegate host and ZooKeeper ensemble node, 198.51.100.136:

```
SERVICED_ISVCS_START=zookeeper
SERVICED_ISVCS_ZOOKEEPER_ID=2
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@198.51.100.135:2888:3888,\
  2@0.0.0.0:2888:3888,3@198.51.100.137:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Delegate host and ZooKeeper ensemble node, 198.51.100.137:

```
SERVICED_ISVCS_START=zookeeper
SERVICED_ISVCS_ZOOKEEPER_ID=3
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@198.51.100.135:2888:3888,\
```

```
  2@198.51.100.136:2888:3888,3@0.0.0.0:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Delegate host, 198.51.100.138:

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
```

Configuring the master host as a ZooKeeper node

This procedure configures the Control Center master host as a node in a ZooKeeper ensemble.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Define the IP address variables for each node in the ZooKeeper ensemble.
   Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts to include in the ensemble:

   ```
   node1=Master
   node2=Delegate-A
   node3=Delegate-B
   ```
3. Set the ZooKeeper node ID to `1`.

   ```
   echo "SERVICED_ISVCS_ZOOKEEPER_ID=1" >> /etc/default/serviced
   ```
4. Specify the nodes in the ZooKeeper ensemble.
   You can copy the following text and paste it in your console:

   ```
   echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
     >> /etc/default/serviced
   ```
5. Specify the nodes in the ZooKeeper quorum.

   ZooKeeper requires a unique quorum definition for each node in its ensemble. To achieve this, replace the IP address or hostname of the master host with `0.0.0.0`.

   You can copy the following text and paste it in your console:

   ```
   q1="1@0.0.0.0:2888:3888"
   q2="2@${node2}:2888:3888"
   q3="3@${node3}:2888:3888"
   echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
     >> /etc/default/serviced
   ```
6. Specify the timeout for inactive connections.
   You can copy the following text and paste it in your console:

   ```
   echo "SERVICED_ZK_SESSION_TIMEOUT=15" >> /etc/default/serviced
   ```
7. Clean up the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.
   c. Locate the original *SERVICED_ZK* variable declaration, and then paste the cut line immediately below it.
   d. Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration at that location.

e Locate the original *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration, and then paste the cut line immediately below it.
f Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration at that location.
g Locate the original *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration, and then paste the cut line immediately below it.
h Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK_SESSION_TIMEOUT* variable declaration at that location.
i Locate the original *SERVICED_ZK_SESSION_TIMEOUT* variable declaration, and then paste the cut line immediately below it.
j Save the file, and then close the editor.

8 Verify the ZooKeeper environment variables.

```
grep -E '^\b*SERVICED' /etc/default/serviced | grep -E '_Z(OO|K)'
```

The following example shows the environment variables for a master host with IP address 198.51.100.135.

Note The value of the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable is formatted to fit the available space. The result of the `grep` command shows the variable and value on the same line.

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_ZOOKEEPER_ID=1
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@0.0.0.0:2888:3888,\
  2@198.51.100.136:2888:3888,3@198.51.100.137:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Configuring delegate host A as a ZooKeeper node

Use this procedure to configure the delegate host designated as *Delegate-A* as a ZooKeeper node.

1 Log in to the delegate host as `root`, or as a user with superuser privileges.
2 Define the IP address variables for each node in the ZooKeeper ensemble.
Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts to include in the ensemble:

```
node1=Master
node2=Delegate-A
node3=Delegate-B
```

3 Set the ID of this node in the ZooKeeper ensemble.

```
echo "SERVICED_ISVCS_ZOOKEEPER_ID=2" >> /etc/default/serviced
```

4 Specify the nodes in the ZooKeeper ensemble.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
  >> /etc/default/serviced
```

5 Specify the nodes in the ZooKeeper quorum.
ZooKeeper requires a unique quorum definition for each node in its ensemble. To achieve this, replace the IP address or hostname of delegate host A with `0.0.0.0`.

You can copy the following text and paste it in your console:

```
q1="1@${node1}:2888:3888"
q2="2@0.0.0.0:2888:3888"
q3="3@${node3}:2888:3888"
echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
  >> /etc/default/serviced
```

6 Specify the timeout for inactive connections.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ZK_SESSION_TIMEOUT=15" >> /etc/default/serviced
```

7 Configure Control Center to start the ZooKeeper service.
You can copy the following text and paste it in your console:

```
echo "SERVICED_ISVCS_START=zookeeper" >> /etc/default/serviced
```

8 Clean up the Control Center configuration file.

a Open `/etc/default/serviced` in a text editor.

b Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.

c Locate the original *SERVICED_ZK* variable declaration, and then paste the cut line immediately below it.

d Comment the original *SERVICED_ZK* declaration, which references only the master host.

Insert the number sign character (`#`) immediately in front of the original *SERVICED_ZK* variable.

e Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration at that location.

f Locate the original *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration, and then paste the cut line immediately below it.

g Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration at that location.

h Locate the original *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration, and then paste the cut line immediately below it.

i Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK_SESSION_TIMEOUT* variable declaration at that location.

j Locate the original *SERVICED_ZK_SESSION_TIMEOUT* variable declaration, and then paste the cut line immediately below it.

k Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_START* variable declaration at that location.

l Locate the original *SERVICED_ISVCS_START* variable declaration, and then paste the cut line immediately below it.

m Save the file, and then close the editor.

9 Verify the ZooKeeper environment variables.

```
grep -E '^\b*SERVICED' /etc/default/serviced \
  | grep -E '(CS_ZO|_ZK|CS_ST)'
```

The following example shows the environment variables for a delegate host with IP address 198.51.100.136.

Note The value of the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable is formatted to fit the available space. The result of the `grep` command shows the variable and value on the same line.

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
```

```
SERVICED_ISVCS_START=zookeeper
SERVICED_ISVCS_ZOOKEEPER_ID=2
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@198.51.100.135:2888:3888,\
  2@0.0.0.0:2888:3888,3@198.51.100.137:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Configuring delegate host B as a ZooKeeper node

Use this procedure to configure the delegate host designated as *Delegate-B* as a ZooKeeper node.

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Define the IP address variables for each node in the ZooKeeper ensemble.
   Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts to include in the ensemble:

   ```
   node1=Master
   node2=Delegate-A
   node3=Delegate-B
   ```
3. Set the ID of this node in the ZooKeeper ensemble.

   ```
   echo "SERVICED_ISVCS_ZOOKEEPER_ID=3" >> /etc/default/serviced
   ```
4. Specify the nodes in the ZooKeeper ensemble.
   You can copy the following text and paste it in your console:

   ```
   echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
     >> /etc/default/serviced
   ```
5. Specify the nodes in the ZooKeeper quorum.

   ZooKeeper requires a unique quorum definition for each node in its ensemble. To achieve this, replace the IP address or hostname of delegate host B with `0.0.0.0`.

   You can copy the following text and paste it in your console:

   ```
   q1="1@${node1}:2888:3888"
   q2="2@${node2}:2888:3888"
   q3="3@0.0.0.0:2888:3888"
   echo "SERVICED_ISVCS_ZOOKEEPER_QUORUM=${q1},${q2},${q3}" \
     >> /etc/default/serviced
   ```
6. Specify the timeout for inactive connections.
   You can copy the following text and paste it in your console:

   ```
   echo "SERVICED_ZK_SESSION_TIMEOUT=15" >> /etc/default/serviced
   ```
7. Configure Control Center to start the ZooKeeper service.
   You can copy the following text and paste it in your console:

   ```
   echo "SERVICED_ISVCS_START=zookeeper" >> /etc/default/serviced
   ```
8. Clean up the Control Center configuration file.
   a. Open `/etc/default/serviced` in a text editor.
   b. Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.
   c. Locate the original *SERVICED_ZK* variable declaration, and then paste the cut line immediately below it.

d Comment the original *SERVICED_ZK* declaration, which references only the master host.

Insert the number sign character (#) immediately in front of the original *SERVICED_ZK* variable.

e Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration at that location.

f Locate the original *SERVICED_ISVCS_ZOOKEEPER_ID* variable declaration, and then paste the cut line immediately below it.

g Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration at that location.

h Locate the original *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable declaration, and then paste the cut line immediately below it.

i Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK_SESSION_TIMEOUT* variable declaration at that location.

j Locate the original *SERVICED_ZK_SESSION_TIMEOUT* variable declaration, and then paste the cut line immediately below it.

k Navigate to the end of the file, and cut the line that contains the *SERVICED_ISVCS_START* variable declaration at that location.

l Locate the original *SERVICED_ISVCS_START* variable declaration, and then paste the cut line immediately below it.

m Save the file, and then close the editor.

9 Verify the ZooKeeper environment variables.

```
grep -E '^\b*SERVICED' /etc/default/serviced \
  | grep -E '(CS_ZO|_ZK|CS_ST)'
```

The following example shows the environment variables for a delegate host with IP address 198.51.100.137.

Note The value of the *SERVICED_ISVCS_ZOOKEEPER_QUORUM* variable is formatted to fit the available space. The result of the `grep` command shows the variable and value on the same line.

```
SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
SERVICED_ISVCS_START=zookeeper
SERVICED_ISVCS_ZOOKEEPER_ID=3
SERVICED_ISVCS_ZOOKEEPER_QUORUM=1@198.51.100.135:2888:3888,\
  2@198.51.100.136:2888:3888,3@0.0.0.0:2888:3888
SERVICED_ZK_SESSION_TIMEOUT=15
```

Importing the ZooKeeper image for Docker

Perform the steps in *Downloading and staging required files* on page 8 before performing this procedure.

Use this procedure to import the Docker image for the ZooKeeper service from a ZooKeeper image file. Repeat this procedure on each node in the ZooKeeper ensemble.

1 Log in to the ZooKeeper ensemble node as `root`, or as a user with superuser privileges.

2 Change directory to `/root`.

```
cd /root
```

3 Extract the ZooKeeper image.

```
./install-zenoss-isvcs-zookeeper-v*.run
```

Image extraction begins when you press the **y** key.

4 Optional: Delete the archive file, if desired.

```
rm -i ./install-zenoss-isvcs-zookeeper-v*.run
```

Starting a ZooKeeper ensemble

Use this procedure to start a ZooKeeper ensemble.

This procedures uses the `nc` utility to query ensemble hosts. If `nc` is not available, you can use `telnet` with *interactive ZooKeeper commands*.

The window of time for starting a ZooKeeper ensemble is relatively short. The goal of this procedure is to restart Control Center on each ensemble node at about the same time, so that each node can participate in electing the leader.

1 Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2 In a separate window, log in to the second node of the ZooKeeper ensemble (*Delegate-A*) as `root`, or as a user with superuser privileges.
3 In a different window, log in to the third node of the ZooKeeper ensemble (*Delegate-B*) as `root`, or as a user with superuser privileges.
4 On all ensemble hosts, stop and start `serviced`.

```
systemctl stop serviced && systemctl start serviced
```

5 On the master host, check the status of the ZooKeeper ensemble.
   a Attach to the container of the ZooKeeper service.

```
docker exec -it serviced-isvcs_zookeeper /bin/bash
```

   b Define IP address variables for each node in the ZooKeeper ensemble.
   Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts in the ensemble:

```
node1=Master
node2=Delegate-A
node3=Delegate-B
```

   c Query the master host and identify its role in the ensemble.

```
{ echo stats; sleep 1; } | nc $node1 2181 | grep Mode
```

   The result includes `leader` or `follower`.
   d Query delegate host A and identify its role in the ensemble.

```
{ echo stats; sleep 1; } | nc $node2 2181 | grep Mode
```

   e Query delegate host B and identify its role in the ensemble.

```
{ echo stats; sleep 1; } | nc $node3 2181 | grep Mode
```

   f Detach from the container of the ZooKeeper service.

```
exit
```

If none of the hosts reports that it is the ensemble leader within a few minutes of starting `serviced`, reboot the hosts.

Updating delegate hosts

The default configuration of delegate hosts sets the value of the *SERVICED_ZK* variable to the master host only. Use this procedure to update the setting to include all of the hosts in the ZooKeeper ensemble. **Perform this procedure on each delegate host that is not a ZooKeeper ensemble node.**

1. Log in to the delegate host as `root`, or as a user with superuser privileges.
2. Define the IP address variables for each node in the ZooKeeper ensemble.
   Replace *Master* with the IP address or hostname of the Control Center master host, and replace *Delegate-A* and *Delegate-B* with the IP addresses or hostnames of the delegate hosts to include in the ensemble:

   ```
   node1=Master
   node2=Delegate-A
   node3=Delegate-B
   ```
3. Specify the nodes in the ZooKeeper ensemble.
   You can copy the following text and paste it in your console:

   ```
   echo "SERVICED_ZK=${node1}:2181,${node2}:2181,${node3}:2181" \
     >> /etc/default/serviced
   ```
4. Update the variable.
   a. Open `/etc/default/serviced` in a text editor.
   b. Navigate to the end of the file, and cut the line that contains the *SERVICED_ZK* variable declaration at that location.
      The value of this declaration specifies three endpoints.
   c. Locate the *SERVICED_ZK* variable near the beginning of the file, and then delete the line it is on.
      The value is just the master host endpoint.
   d. Paste the *SERVICED_ZK* variable declaration from the end of the file in the location of the just-deleted declaration.
   e. Save the file, and then close the editor.
5. Verify the setting.

   ```
   grep -E '^\b*SERVICED_ZK' /etc/default/serviced
   ```
   The following example shows the environment variable for a delegate host that is not a node in the ZooKeeper ensemble:

   ```
   SERVICED_ZK=198.51.100.135:2181,198.51.100.136:2181,198.51.100.137:2181
   ```
6. Restart Control Center.

   ```
   systemctl restart serviced
   ```

Starting and stopping Control Center deployments

A

This appendix includes procedures for stopping and starting both single-host and multi-host Control Center deployments.

Note The procedures in this appendix assume that Control Center is the only source of Docker containers that are run on a host.

Stopping Control Center (single-host deployment)

Use this procedure to stop the Control Center service (`serviced`) in a single-host deployment.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Stop the top-level service `serviced` is managing, if necessary.
   a. Show the status of running services.

      ```
      serviced service status
      ```

      The top-level service is the service listed immediately below the headings line.

      - If the status of the top-level service and all child services is `stopped`, proceed to the next step.
      - If the status of the top-level service and all child services is **not** `stopped`, perform the remaining substeps.

   b. Stop the top-level service.
      Replace *Service* with the name or identifier of the top-level service:

      ```
      serviced service stop Service
      ```

   c. Monitor the stop.

      ```
      serviced service status
      ```

      When the status of the top-level service and all child services is `stopped`, proceed to the next step.
3. Stop the Control Center service.

   ```
   systemctl stop serviced
   ```

4. Ensure that no containers remain in the local repository.

a Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the following substeps.

b Remove all remaining containers.

```
docker ps -qa | xargs --no-run-if-empty docker rm -fv
```

c Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the remaining substeps.

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

f Log in to the master host as `root`, or as a user with superuser privileges.

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Starting Control Center (single-host deployment)

Use this procedure to start Control Center in a single-host deployment. The default configuration of the Control Center service (`serviced`) is to start when the host starts. This procedure is only needed after stopping `serviced` to perform maintenance tasks.

1 Log in to the master host as `root`, or as a user with superuser privileges.

2 Determine whether `serviced` is configured to start when the system starts.

```
systemctl is-enabled serviced
```

- If the result is `enabled`, proceed to the next step.
- If the result is `disabled`, enter the following command:

```
systemctl enable serviced
```

3 Start the Control Center service.

```
systemctl start serviced
```

4 Optional: Monitor the startup, if desired.

```
journalctl -u serviced -f -o cat
```

Once Control Center is started, it is ready to start managing applications. For more information, refer to the documentation of your application.

Stopping Control Center (multi-host deployment)

To stop Control Center in a multi-host deployment, perform the procedures in this section, in order.

Stopping a master host (multi-host deployment)

Use this procedure to stop the Control Center service (`serviced`) on the master host in a multi-host deployment.

1 Log in to the master host as `root`, or as a user with superuser privileges.
2 Stop the top-level service `serviced` is managing, if necessary.
   a Show the status of running services.

```
serviced service status
```

   The top-level service is the service listed immediately below the headings line.

   - If the status of the top-level service and all child services is `stopped`, proceed to the next step.
   - If the status of the top-level service and all child services is **not** `stopped`, perform the remaining substeps.

   b Stop the top-level service.
   Replace *Service* with the name or identifier of the top-level service:

```
serviced service stop Service
```

   c Monitor the stop.

```
serviced service status
```

   When the status of the top-level service and all child services is `stopped`, proceed to the next step.
3 Stop the Control Center service.

```
systemctl stop serviced
```

4 Ensure that no containers remain in the local repository.
   a Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

   - If the command returns no result, stop. This procedure is complete.
   - If the command returns a result, perform the following substeps.

   b Remove all remaining containers.

```
docker ps -qa | xargs --no-run-if-empty docker rm -fv
```

c Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, stop. This procedure is complete.
- If the command returns a result, perform the remaining substeps.

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

f Log in to the master host as `root`, or as a user with superuser privileges.

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Stopping a delegate host

Use this procedure to stop the Control Center service (`serviced`) on a delegate host in a multi-host deployment. Repeat this procedure on each delegate host in your deployment.

1 Log in to the delegate host as `root`, or as a user with superuser privileges.

2 Stop the Control Center service.

```
systemctl stop serviced
```

3 Ensure that no containers remain in the local repository.

a Display the identifiers of all containers, running and exited.

```
docker ps -qa
```

- If the command returns no result, proceed to the next step.
- If the command returns a result, perform the following substeps.

b Remove all remaining containers.

```
docker ps -qa | xargs --no-run-if-empty docker rm -fv
```

- If the remove command completes, proceed to the next step.
- If the remove command does not complete, the most likely cause is an NFS conflict. Perform the following substeps.

c Stop the NFS and Docker services.

```
systemctl stop nfs && systemctl stop docker
```

d Start the NFS and Docker services.

```
systemctl start nfs && systemctl start docker
```

e Repeat the attempt to remove all remaining containers.

```
docker ps -qa | xargs --no-run-if-empty docker rm -fv
```

- If the remove command completes, proceed to the next step.
- If the remove command does not complete, perform the remaining substeps.

f Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

g Reboot the host.

```
reboot
```

h Log in to the delegate host as `root`, or as a user with superuser privileges.

i Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

4 Dismount all filesystems mounted from the Control Center master host.

This step ensures no stale mounts remain when the storage on the master host is replaced.

a Identify filesystems mounted from the master host.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs'
```

- If the preceding command returns no result, stop. This procedure is complete.
- If the preceding command returns a result, perform the following substeps.

b Force the filesystems to dismount.

```
for FS in $(awk '/serviced/ { print $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs')
do
  umount -f $FS
done
```

c Identify filesystems mounted from the master host.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs'
```

- If the preceding command returns no result, stop. This procedure is complete.
- If the preceding command returns a result, perform the following substeps.

d Perform a lazy dismount.

```
for FS in $(awk '/serviced/ { print $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs')
do
  umount -f -l $FS
done
```

e Restart the NFS service.

```
systemctl restart nfs
```

f Determine whether any filesystems remain mounted from the master host.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts \
  | grep -v '/opt/serviced/var/isvcs'
```

- If the preceding command returns no result, stop. This procedure is complete.
- If the preceding command returns a result, perform the remaining substeps.

g Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

h Reboot the host.

```
reboot
```

i Log in to the delegate host as `root`, or as a user with superuser privileges.

j Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Starting Control Center (multi-host deployment)

Use this procedure to start Control Center in a multi-host deployment. The default configuration of the Control Center service (`serviced`) is to start when the host starts. This procedure is only needed after stopping `serviced` to perform maintenance tasks.

1 Log in to the master host as `root`, or as a user with superuser privileges.

2 Determine whether `serviced` is configured to start when the system starts.

```
systemctl is-enabled serviced
```

- If the result is `enabled`, proceed to the next step.
- If the result is `disabled`, enter the following command:

```
systemctl enable serviced
```

3 Identify the hosts in the ZooKeeper ensemble.

```
grep -E '^\b*SERVICED_ZK=' /etc/default/serviced
```

The result is a list of 1, 3, or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.

4 In separate windows, log in to each of the delegate hosts that are nodes in the ZooKeeper ensemble as `root`, or as a user with superuser privileges.

5 On all ensemble hosts, start `serviced`.

The window of time for starting a ZooKeeper ensemble is relatively short. The goal of this step is to start Control Center on each ensemble node at about the same time, so that each node can participate in electing the leader.

```
systemctl start serviced
```

6 On the master host, check the status of the ZooKeeper ensemble.

a Attach to the container of the ZooKeeper service.

```
docker exec -it serviced-isvcs_zookeeper bash
```

b Query the master host and identify its role in the ensemble.
Replace *Master* with the hostname or IP address of the master host:

```
{ echo stats; sleep 1; } | nc Master 2181 | grep Mode
```

The result includes `leader` or `follower`. When multiple hosts rely on the ZooKeeper instance on the master host, the result includes `standalone`.

c Query the other delegate hosts to identify their role in the ensemble.
Replace *Delegate* with the hostname or IP address of a delegate host:

```
{ echo stats; sleep 1; } | nc Delegate 2181 | grep Mode
```

d Detach from the container of the ZooKeeper service.

```
exit
```

If none of the nodes reports that it is the ensemble leader within a few minutes of starting `serviced`, reboot the ensemble hosts.

7 Log in to each of the delegate hosts that are not nodes in the ZooKeeper ensemble as `root`, or as a user with superuser privileges, and then start `serviced`.

```
systemctl start serviced
```

8 Optional: Monitor the startup, if desired.

```
journalctl -u serviced -f -o cat
```

Once Control Center is started, it is ready to start managing applications. For more information, refer to the documentation of your application.

Storage management utility B

This appendix includes a description of the `serviced-storage` command, the required utility for creating the Docker thin pool and creating and managing the Control Center application data thin pool.

serviced-storage

The `serviced-storage` command manages Control Center storage.

Use this command to create LVM thin pools for Docker and Control Center.

USAGE

```
serviced-storage  [-h|--help] [-o DeviceMapperOption=Value] \
  [-v] Command [CommandOptions]
```

GLOBAL OPTIONS

--help, -h

Shows the help information.

-o *DeviceMapperOption=Value*

A device mapper option. Applies only to device mapper drivers.

-v

Displays verbose logging.

COMMANDS

check

Check for orphaned devices.

create

Create a volume on a driver.

create-thin-pool

Create an LVM thin pool.

disable

Disable a driver.

init

Initialize a driver.

list
Print volumes on a driver.

mount
Mount an existing volume from a driver.

remove
Remove an existing volume from a driver.

resize
Resize an existing volume.

set
Set the default driver.

status
Print the driver status

sync
Sync data from a volume to another volume.

unset
Unset the default driver.

version
Print the version and exit.

serviced-storage check

The `serviced-storage check` command searches for orphaned snapshot devices in the `serviced` application data thin pool and removes them, if requested. This command requires the path of `serviced` tenant volumes, which is determined by the *SERVICED_VOLUMES_PATH* variable in `/etc/default/serviced`. The default path is `/opt/serviced/var/volumes`.

Syntax:

```
serviced-storage [GlobalOptions] check [-c|--clean] Path
```

Command options:

[-c|--clean]
Remove orphaned snapshot devices.

serviced-storage create-thin-pool

The `serviced-storage create-thin-pool` command creates an LVM thin pool either for Docker data or for Control Center application data. When devices are specified, the command creates an LVM volume group.

Syntax:

```
serviced-storage [GlobalOptions] create-thin-pool \
  [-s|--size]=[Value][G|%] [docker|serviced] \
  [DevicePath [DevicePath...]|VolumeGroupName]
```

Command options:

[-s|--size]=[*Value*][G|%]

The size of the thin pool to create. The size can be a fixed value (in gigabytes) or a relative value (a percentage) of the available storage. When this option is not used, the thin pool size defaults to 90% of the specified storage resource.

serviced-storage resize

The `serviced-storage resize` command increases the size of a `serviced` tenant device in its LVM thin pool. Like LVM thin pools, the size of a `serviced` tenant device can never decrease.

Syntax:

```
serviced-storage [GlobalOptions] resize \
  [-d|--driver]=Value TenantID NewSize
```

Command options:

[-d|--driver]=*Value*

The path of the tenant volume.

EXAMPLES

Create an LVM volume group named `zenoss` and use it for both thin pools:

```
vgcreate zenoss /dev/sdb /dev/sdc
serviced-storage create-thin-pool --size=50G docker zenoss
serviced-storage create-thin-pool --size=50% serviced zenoss
```

If you specify devices or partitions, `serviced-storage` creates an LVM volume group with the same name as the thin pool. The following example yields the same result as the previous, except the name of the volume group is `docker` instead of `zenoss`:

```
serviced-storage create-thin-pool docker /dev/sdb /dev/sdc
serviced-storage create-thin-pool serviced docker
```

Create thin pools on separate block devices:

```
serviced-storage create-thin-pool docker /dev/sdb
serviced-storage create-thin-pool serviced /dev/sdc
```

Create thin pools on separate partitions:

```
serviced-storage create-thin-pool docker /dev/sdb1
serviced-storage create-thin-pool serviced /dev/sdc3
```

Increase the size of the `serviced` LVM thin pool, and then increase the size of a `serviced` tenant device.

```
lvextend -L+300G zenoss/serviced-pool
serviced-storage -o dm.thinpooldev=/dev/mapper/zenoss-serviced--pool \
  resize -d /opt/serviced/var/volumes 58uuetj38draeu9alp6002b1y 200G
```

Identify the `serviced` application data thin pool, and then remove orphaned snapshot devices.

```
ls /dev/mapper | grep serviced
serviced-storage -o dm.thinpooldev=/dev/mapper/zenoss-serviced--pool \
  check -c /opt/serviced/var/volumes
```

Control Center configuration variables C

This appendix includes recommended best practices for editing the `serviced` configuration file, and descriptions of the variables in the file.

Best practices for configuration files

The Control Center configuration file, `/etc/default/serviced`, contains Bash environment variables that are read by the `serviced` daemon startup script. The following list describes recommended best practices for its use and maintenance:

1. When in doubt, make a backup. Before editing, making a backup copy of the configuration file is always the safest choice.
2. Copy a variable, then edit the copy. If you need to revert a variable to its default value, you don't have to leave the file to look it up.
3. Copy and edit a variable only if the default value needs to be changed. It's easier to troubleshoot problems when only non-default variables are copied and edited.
4. Put the first character of the variable declaration in the first column of its line. It's easier to `grep` for settings when each one starts a line.
5. Add customizations to the top of the file. Customizations at the end of the file or scattered throughout the file may be overlooked.
6. In high-availability deployments, the contents of `/etc/default/serviced` on the master nodes must be identical. Use a utility like `sum` to compare the files quickly.

Control Center configuration file

The Control Center configuration file, `/etc/default/serviced`, contains Bash environment variables that are read by the `serviced` daemon startup script. The order of the following list matches the order of the variables in the file.

HOME

Default: (the value of shell variable *HOME*)

The path Docker clients use to locate the `.docker/config.json` authentication file, which contains Docker Hub credentials.

TMPDIR

Default: (the value of shell variable *TMPDIR*)

The path `serviced` uses for temporary files.

GOMAXPROCS

Default: `2`

The maximum number of CPU cores `serviced` uses.

SERVICED_MASTER

Default: `1` (true)

Assigns the role of a `serviced` instance, either master or delegate. The master runs the application services scheduler and other internal services. Delegates run the application services assigned to the resource pool to which they belong.

Only one `serviced` instance can be the master; all other instances must be delegates. The default value assigns the master role. To assign the delegate role, set the value to `0` (false). This variable must be explicitly set on all Control Center hosts.

SERVICED_MASTER_IP

Default: `127.0.0.1`

A convenience variable, for use in places where the IP address or hostname of the master host is required. This variable is unused unless it is both set here and referenced elsewhere. (For example, by replacing `{{SERVICED_MASTER_IP}}` with `$SERVICED_MASTER_IP`.)

SERVICED_MASTER_POOLID

Default: `default`

The name of the default resource pool. This variable is only used the first time `serviced` is started.

SERVICED_ZK

Default: (none)

The list of endpoints in the `serviced` ZooKeeper ensemble, separated by the comma character (`,`). Each endpoint identifies an ensemble node. Each Control Center server and in-container proxy uses *SERVICED_ZK* to create a randomized, round-robin list, and cycles through the list when it attempts to establish a connection with the lead ZooKeeper host.

SERVICED_DOCKER_REGISTRY

Default: `localhost:5000`

The endpoint of the local Docker registry, which `serviced` uses to store internal services and application images.

If the default value is changed, the host's Docker configuration file must include the `--insecure-registry` flag with the same value as this variable.

The safest replacement for `localhost` is the IPv4 address of the registry host. Otherwise, the fully-qualified domain name of the host must be specified.

SERVICED_OUTBOUND_IP

Default: (none)

The IPv4 address that delegates use to connect to the master host. When no address is specified, `serviced` attempts to discover its public IP address by pinging `google.com`.

This variable must be set on all Control Center hosts in either of the following scenarios:

- Control Center is deployed behind a firewall and `google.com` is not reachable. Set the value to the IPv4 address of the master host.
- Control Center is deployed in a high-availability cluster. Set the value to the virtual IPv4 address of the high-availability cluster (*HA-Virtual-IP*).

Note Setting the Docker *HTTP_PROXY* or *HTTPS_PROXY* environment variables prevents access to the IP address defined with this variable. To enable access, unset the Docker variables, and then reboot the host.

SERVICED_STATIC_IPS

Default: (none)

A list of one or more static IP addresses that are available for IP assignment. Use the comma character (`,`) to separate addresses.

SERVICED_ENDPOINT

Default: `{{SERVICED_MASTER_IP}}:4979`

The endpoint of the `serviced` RPC server. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host. The port number of this endpoint must match the value of the *SERVICED_RPC_PORT* variable defined on the `serviced` master host.

SERVICED_MAX_RPC_CLIENTS

Default: `3`

The preferred maximum number of simultaneous connections a `serviced` delegate uses for RPC requests. The value is used to create a pool of sockets, which are reused as needed. Increasing the value increases the number of open sockets and the use of socket-related operating system resources.

When the demand for connections exceeds the supply of open sockets, `serviced` opens more sockets. When demand eases, `serviced` reduces the number of open sockets to the preferred maximum.

SERVICED_RPC_PORT

Default: `4979`

The port on which the `serviced` RPC server listens for connections. The value of this variable must match the port number defined for the *SERVICED_ENDPOINT* variable on all `serviced` delegate hosts.

SERVICED_RPC_CERT_VERIFY

Default: `false`

Determines whether `serviced` performs TLS certificate verification for RPC connections. The certificate is defined by the *SERVICED_CERT_FILE* variable.

SERVICED_RPC_DISABLE_TLS

Default: `false`

Determines whether `serviced` encrypts RPC traffic with TLS.

SERVICED_RPC_TLS_MIN_VERSION

Default: `VersionTLS10`

The minimum version of TLS `serviced` accepts for RPC connections. Valid values include the default, `VersionTLS11`, and `VersionTLS12`.

SERVICED_RPC_TLS_CIPHERS

Default: (list of ciphers)

The list of TLS ciphers `serviced` prefers for RPC connections, separated by the comma character (`,`):

- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

Other ciphers are supported; the preceding ciphers provide strong security for relatively low processing overhead.

An instance of `serviced` is on both ends of an RPC connection, so both daemons use the first cipher in the list. To use a different cipher, put it first in the list, on all Control Center hosts.

SERVICED_UI_PORT

Default: `:443`

The port on which the `serviced` HTTP server listens for requests for its internal services and for tenant services. The value may be expressed as follows:

IP-Address:*Port-Number*
:*Port-Number*
Port-Number

Tenant applications can specify alternative ports with the port public endpoint feature.

The value of this variable must be identical on all Control Center hosts in a deployment.

SERVICED_UI_POLL_FREQUENCY

Default: `3`

The number of seconds between polls from Control Center browser interface clients. The value is included in a JavaScript library that is sent to the clients.

SERVICED_MUX_PORT

Default: `22250`

The port `serviced` uses for traffic among Docker containers.

SERVICED_MUX_DISABLE_TLS

Default: `0`

Determines whether inter-host traffic among Docker containers is encrypted with TLS. Intra-host traffic among Docker containers is not encrypted. To disable encryption, set the value to `1`.

SERVICED_MUX_TLS_MIN_VERSION

Default: `VersionTLS10`

The minimum version of TLS `serviced` accepts for mux traffic. Valid values include the default, `VersionTLS11`, and `VersionTLS12`.

SERVICED_MUX_TLS_CIPHERS

Default: (list of ciphers)

The list of TLS ciphers `serviced` prefers for mux traffic, separated by the comma character (`,`):

- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

Other ciphers are supported; the preceding ciphers provide strong security for relatively low processing overhead.

An instance of `serviced` is on both ends of a mux connection, so both daemons use the first cipher in the list. To use a different cipher, put it first in the list, on all Control Center hosts.

SERVICED_ISVCS_PATH

Default: `/opt/serviced/var/isvcs`

The location of `serviced` internal services data.

SERVICED_VOLUMES_PATH

Default: `/opt/serviced/var/volumes`

The location of `serviced` application data.

SERVICED_BACKUPS_PATH

Default: `/opt/serviced/var/backups`

The location of `serviced` backup files.

SERVICED_LOG_PATH

Default: `/var/log/serviced`

The location of `serviced` audit log files. Non-audit (operations) messages are written to `journald`.

SERVICED_KEY_FILE

Default: `$TMPDIR/zenoss_key.[0-9]+`

The path of a digital certificate key file. Choose a location that is not modified during operating system updates, such as `/etc`.

This key file is used for all TLS-encrypted communications (RPC, mux, and HTTP). The default, insecure key file is created when the `serviced` web server first starts, and is based on a public key that is compiled into `serviced`.

SERVICED_CERT_FILE

Default: `$TMPDIR/zenoss_cert.[0-9]+`

The path of a digital certificate file. Choose a location that is not modified during operating system updates, such as `/etc`. Certificates with passphrases are not supported.

This certificate file is used for all TLS-encrypted communications (RPC, mux, and HTTP). The default, insecure certificate file is created when the `serviced` web server first starts, and is based on a public certificate that is compiled into `serviced`.

SERVICED_TLS_MIN_VERSION

Default: `VersionTLS10`

The minimum version of TLS that `serviced` accepts for HTTP traffic. Valid values include the default, `VersionTLS11`, and `VersionTLS12`.

SERVICED_TLS_CIPHERS

Default: (list of ciphers)

The list of TLS ciphers that `serviced` accepts for HTTP traffic, separated by the comma character (`,`):

1. `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
2. `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
3. `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
4. `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
5. `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
6. `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
7. `TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA`
8. `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA`
9. `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
10. `TLS_RSA_WITH_AES_256_CBC_SHA`
11. `TLS_RSA_WITH_AES_128_CBC_SHA`
12. `TLS_RSA_WITH_3DES_EDE_CBC_SHA`
13. `TLS_RSA_WITH_AES_128_GCM_SHA256`
14. `TLS_RSA_WITH_AES_256_GCM_SHA384`

To disable support for most ciphers, you can remove them from the list. The following rules apply to the list:

- The first cipher, `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`, must always be present in the list of ciphers.
- The first four ciphers in the list must always precede any of the ciphers that appear after the first four. The first four ciphers are valid for HTTP/2, while the remaining ciphers are not.

SERVICED_FS_TYPE

Default: `devicemapper`

The driver to manage application data storage on the `serviced` master host. Only `devicemapper` is supported in production deployments.

The only supported storage layout for the `devicemapper` driver is an LVM thin pool. To create a thin pool, use the `serviced-storage` utility. To specify the name of the thin pool device, use the *SERVICED_DM_THINPOOLDEV* variable.

SERVICED_DM_ARGS

Default: (none)

Customized startup arguments for the `devicemapper` storage driver.

SERVICED_DM_BASESIZE

Default: `100G`

The base size of virtual storage devices for tenants in the application data thin pool, in gigabytes. The units symbol (`G`) is required. This variable is used when `serviced` starts for the first time, to set the initial size of tenant devices, and when a backup is restored, to set the size of the restored tenant device.

The base size device is sparse device that occupies at most 1MB of space in the application data thin pool; its size has no immediate practical impact. However, the application data thin pool should have enough space for twice the size of each tenant device it supports, to store both the data itself and snapshots of the data. Since the application data thin pool is an LVM logical volume, its size can be increased at any time. Likewise, the size of a tenant device can be increased, as long as the available space in the thin pool can support the larger tenant device plus snapshots.

SERVICED_DM_LOOPDATASIZE

Default: `100G`

Specifies the size of the data portion of the loop-back file. This setting is ignored when *SERVICED_ALLOW_LOOP_BACK* is `false`.

SERVICED_DM_LOOPMETADATASIZE

Default: `2G`

Specifies the size of the metadata portion of the loop-back file. This setting is ignored when *SERVICED_ALLOW_LOOP_BACK* is `false`.

SERVICED_DM_THINPOOLDEV

Default: (none)

The name of the thin pool device to use with the `devicemapper` storage driver.

SERVICED_STORAGE_STATS_UPDATE_INTERVAL

Default: `300` (5 minutes)

The number of seconds between polls of kernel statistics about the application data thin pool.

This setting is ignored when the operating system kernel version is less than 3.10.0-366.

SERVICED_ALLOW_LOOP_BACK

Default: `false`

Determines whether loop-back files can be used with the `devicemapper` storage driver. This option is not supported for production use.

SERVICED_MAX_CONTAINER_AGE

Default: `86400` (24 hours)

The number of seconds `serviced` waits before removing a stopped container.

SERVICED_VIRTUAL_ADDRESS_SUBNET

Default: `10.3.0.0/16`

The private subnet for containers that use virtual IP addresses on a host. This value may be unique on each Control Center host, if necessary.

RFC 1918 restricts private networks to the 10.0/24, 172.16/20, and 192.168/16 address spaces. However, `serviced` accepts any valid IPv4 address space.

Specify the value in CIDR notation. A `/29` network provides sufficient address space.

SERVICED_LOG_LEVEL

Default: `0`

The log level `serviced` uses when writing to the system log. Valid values are `0` (normal) and `2` (debug).

SERVICED_LOG_ADDRESS

Default: `{{SERVICED_MASTER_IP}}:5042`

The endpoint of the logstash service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

SERVICED_LOGSTASH_ES

Default: `{{SERVICED_MASTER_IP}}:9100`

The endpoint of the Elasticsearch service for logstash. On delegate hosts, replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the Elasticsearch host, which by default is the `serviced` master host.

SERVICED_LOGSTASH_MAX_DAYS

Default: `14`

The maximum number of days to keep application logs in the logstash database before purging them.

SERVICED_LOGSTASH_MAX_SIZE

Default: `10`

The maximum size of the logstash database, in gigabytes.

SERVICED_LOGSTASH_CYCLE_TIME

Default: `6`

The amount of time between logstash purges, in hours.

SERVICED_STATS_PORT

Default: `{{SERVICED_MASTER_IP}}:8443`

The endpoint of the `serviced` metrics consumer service. Replace `{{SERVICED_MASTER_IP}}` with the IP address or hostname of the `serviced` master host.

SERVICED_STATS_PERIOD

Default: `10`

The frequency, in seconds, at which delegates gather metrics to send to the `serviced` metrics consumer service on the master host.

SERVICED_SVCSTATS_CACHE_TIMEOUT

Default: `5`

The number of seconds to cache statistics about services. The cache is used by Control Center browser interface clients.

SERVICED_DEBUG_PORT

Default: `6006`

The port on which `serviced` listens for HTTP requests for the *Go profiler*. To stop listening for requests, set the value to `-1`.

SERVICED_ISVCS_ENV_[0-9]+

Default: (none)

Startup arguments to pass to internal services. You may define multiple arguments, each for a different internal service. The variables themselves, and their arguments, use the following syntax:

`SERVICED_ISVCS_ENV_`***`%d`***

Each variable name ends with a unique integer in place of *%d*.

`Service-Name:Key=Value`

The value of each variable includes the following elements, in order:

1. *Service-Name*, the internal service name. The following command returns the internal service names that may be used for *Service-Name*:

   ```
   docker ps | awk '/serviced-isvcs:/{print $NF}'
   ```

2. The colon character (`:`).
3. *Key*, a variable to pass to the internal service.
4. The equals sign character (=).
5. *Value*, the definition of the variable to pass to the internal service.

The following example variable passes `ES_JAVA_OPTS=-Xmx4g` to the Elasticsearch internal service.

```
SERVICED_ISVCS_ENV_0=serviced-isvcs_elasticsearch-
logstash:ES_JAVA_OPTS=-Xmx4g
```

SERVICED_ADMIN_GROUP

Default: `wheel`

The name of the Linux group on the `serviced` master host whose members are authorized to use the `serviced` browser interface. You may replace the default group with a group that does not have superuser privileges.

SERVICED_ALLOW_ROOT_LOGIN

Default: `1` (true)

Determines whether the `root` user account on the `serviced` master host may be used to gain access to the `serviced` browser interface.

SERVICED_IPTABLES_MAX_CONNECTIONS

Default: `655360`

The default value of this variable ensures that a `serviced` delegate does not run out of connections if the `serviced` master goes down. The connections are automatically cleaned up by the kernel soon after the `serviced` master comes back online.

SERVICED_SNAPSHOT_TTL

Default: `12`

The number of hours an application data snapshot is retained before removal. To disable snapshot removal, set the value to zero. The application data storage can fill up rapidly when this value is zero or too high.

SERVICED_NFS_CLIENT

Default: `1`

DEPRECATED: Prevent a delegate host from mounting the DFS.

SERVICED_SERVICE_MIGRATION_TAG

Default: `1.0.2`

Overrides the default value for the service migration image.

SERVICED_ISVCS_START

Default: (none)

Enables one or more internal services to run on a delegate host. Currently, only `zookeeper` has been tested.

SERVICED_ISVCS_ZOOKEEPER_ID

Default: (none)

The unique identifier of a ZooKeeper ensemble node. The identifier must be a positive integer.

SERVICED_ISVCS_ZOOKEEPER_QUORUM

Default: (none)

The comma-separated list of nodes in a ZooKeeper ensemble. Each entry in the list specifies the ZooKeeper ID, IP address or hostname, peer communications port, and leader communications port of a node in the ensemble. Each quorum definition must be unique, so the IP address or hostname of the "current" host must be 0.0.0.0.

The following example shows the syntax of a node entry:

```
ZooKeeper-ID@Host-IP-Or-Name:2888:3888
```

SERVICED_DOCKER_LOG_DRIVER

Default: `json-file`

The log driver for all Docker container logs, including containers for Control Center internal services. Valid values:

- `json-file`
- `syslog`
- `journald`
- `gelf`
- `fluentd`
- `none`

This is a direct port of the Docker `--log-driver` option.

SERVICED_DOCKER_LOG_CONFIG

Default: `max-file=5,max-size=10m`

A comma-separated list of Docker `--log-opt` options as *`key=value`* pairs. To specify the default values for a log driver, or for drivers that need no additional options, such as `journald`, use a single comma character (`,`) as the value of this variable.

SERVICED_DOCKER_DNS

Default: (empty)

The IP address of one or more DNS servers. The value of this variable is injected into each Docker container that `serviced` starts. Separate multiple values with the comma character (`,`).

SERVICED_OPTS

Default: (empty)

Special options for the `serviced` startup command.

SERVICED_SNAPSHOT_USE_PERCENT

Default: `20`

The amount of free space in the thin pool specified with *SERVICED_DM_THINPOOLDEV*, expressed as a percentage the total size. This value is used to determine whether the thin pool can hold a new snapshot.

SERVICED_ZK_SESSION_TIMEOUT

Default: `15`

The number of seconds the lead ZooKeeper host waits before flushing an inactive connection.

SERVICED_ZK_CONNECT_TIMEOUT

Default: `1`

The number of seconds Control Center waits for a connection to the lead ZooKeeper host.

SERVICED_ZK_PER_HOST_CONNECT_DELAY

Default: `0`

The number of seconds Control Center waits before attempting to connect to the next host in its round-robin list of ZooKeeper hosts. For more information about the round-robin list, see *SERVICED_ZK*.

SERVICED_ZK_RECONNECT_START_DELAY

Default: `1`

SERVICED_ZK_RECONNECT_START_DELAY and *SERVICED_ZK_RECONNECT_MAX_DELAY* are used together when Control Center is unable to re-establish a connection with the lead ZooKeeper host.

To prevent unnecessary spikes in TCP traffic, Control Center waits a randomized amount of time that is equal to plus or minus 20% of the value of *SERVICED_ZK_RECONNECT_START_DELAY*. If Control Center is unable to reconnect after contacting all of the hosts in its round-robin list of ZooKeeper hosts, the wait time is increased by a randomized value and the process of attempting to reconnect begins again. If the attempts fail again, the process repeats until the wait time reaches the value of *SERVICED_ZK_RECONNECT_MAX_DELAY*, and the wait time of subsequent reconnection attempts is capped at *SERVICED_ZK_RECONNECT_MAX_DELAY*. Once connection is re-established, the wait time is reset to *SERVICED_ZK_RECONNECT_START_DELAY*.

For more information about the round-robin list, see *SERVICED_ZK*.

SERVICED_ZK_RECONNECT_MAX_DELAY

Default: `1`

See *SERVICED_ZK_RECONNECT_START_DELAY*.

SERVICED_ES_STARTUP_TIMEOUT

Default: `240`

The number of seconds to wait for the Elasticsearch service to start.

SERVICED_MAX_DFS_TIMEOUT

Default: `300`

The number of seconds until a DFS snapshot attempt times out.

SERVICED_RPC_DIAL_TIMEOUT

Default: `30`

The number of seconds until an RPC connection attempt times out.

SERVICED_AUTH_TOKEN_EXPIRATION

Default: `3600` (1 hour)

The expiration time, in seconds, of delegate authentication tokens. This timeout affects RPC, mux, and `serviced` internal services endpoint communications.

SERVICED_CONTROLLER_BINARY

Default: `/opt/serviced/bin/serviced-controller`

The path of the `serviced-controller` binary, which runs in every container that `serviced` manages.

SERVICED_HOME

Default: `/opt/serviced`

The path of the home directory for `serviced`.

SERVICED_ETC_PATH

Default: `/opt/serviced/etc`

The path of the directory for `serviced` configuration files. The default is *`SERVICED_HOME`*`/etc`.

SERVICED_VHOST_ALIASES

Default: (none)

A list of hostname aliases for a host; for example, `localhost`. Separate multiple values with the comma character (`,`).

Configuring a private master NTP server

D

Control Center requires a common time source. The procedures in this appendix configure a private master *NTP* server to synchronize the system clocks of all Control Center hosts.

A private master server is only required when a multi-host deployment does not have internet access and no time synchronization servers are available behind the firewall. Single-host deployments do not require time synchronization, and deployments with internet access can rely on the default public time servers that are configured in `/etc/ntp.conf`.

Note VMware vSphere guest systems can synchronize their system clocks with the host system. If that feature is enabled, it must be disabled to configure a private master NTP server or to use a time synchronization server that is available behind the firewall. For more information, refer to the VMware documentation for your version of vSphere.

The procedures in this appendix assume that Control Center is not installed. If it is installed, stop the `serviced` service before configuring NTP.

Configuring an NTP master server

Use this procedure to configure an NTP master server on the Control Center master host. Perform this procedure only if the host does not have internet access.

Note On VMware vSphere guests, before performing this procedure, disable time synchronization between guest and host operating systems.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Create a backup of the NTP configuration file.

```
cp -p /etc/ntp.conf /etc/ntp.conf.orig
```

3. Edit the NTP configuration file.
   a. Open `/etc/ntp.conf` with a text editor.
   b. Replace all of the lines in the file with the following lines:

```
# Use the local clock
server 127.127.1.0 prefer
fudge  127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
```

```
broadcastdelay 0.008

# Give localhost full access rights
restrict 127.0.0.1

# Grant access to client hosts
restrict Address-Range mask Netmask nomodify notrap
```

c. Replace *Address-Range* with the range of IPv4 network addresses that are allowed to query this NTP server.

   For example, the following IP addresses are assigned to Control Center hosts:

   ```
   203.0.113.10
   203.0.113.11
   203.0.113.12
   203.0.113.13
   ```

   For the preceding addresses, the value for *Address-Range* is `203.0.113.0`.

d. Replace *Netmask* with the IPv4 network mask that corresponds with the address range.
   For example, a valid network mask for `203.0.113.0` is `255.255.255.0`.

e. Save the file and exit the editor.

4. Stop Control Center.

```
systemctl stop serviced
```

5. Enable and start the NTP daemon.

   a. Enable the `ntpd` daemon.

   ```
   systemctl enable ntpd
   ```

   b. Configure `ntpd` to start when the system starts.

   Currently, an unresolved issue associated with NTP prevents `ntpd` from restarting correctly after a reboot, and the following commands provide a workaround to ensure that it does.

   ```
   echo "systemctl start ntpd" >> /etc/rc.d/rc.local
   chmod +x /etc/rc.d/rc.local
   ```

   c. Start `ntpd`.

   ```
   systemctl start ntpd
   ```

6. Start Control Center.

```
systemctl start serviced
```

Configuring NTP clients

Use this procedure to configure a delegate hosts to synchronize its clocks with the NTP server on the Control Center master host. Perform this procedure only if the hosts do not have internet access. Repeat this procedure on each Control Center delegate host.

Note On VMware vSphere guests, before performing this procedure, disable time synchronization between guest and host operating systems.

1. Log in to the Control Center delegate host as `root`, or as a user with superuser privileges.
2. Create a backup of the NTP configuration file.

```
cp -p /etc/ntp.conf /etc/ntp.conf.orig
```

3. Edit the NTP configuration file./
   a. Open `/etc/ntp.conf` with a text editor.
   b. Replace all of the lines in the file with the following lines:

```
# Point to the master time server
server Master-Address

restrict default ignore
restrict 127.0.0.1
restrict Master-Address mask 255.255.255.255 nomodify notrap noquery

driftfile /var/lib/ntp/drift
```

   c. Replace both instances of *Master-Address* with the IPv4 address of the host where the NTP server is running (the Control Center master host).
   d. Save the file and exit the editor.
4. Stop Control Center.

```
systemctl stop serviced
```

5. Synchronize the clock with the master server.

```
ntpd -gq
```

6. Enable and start the NTP daemon.
   a. Enable the `ntpd` daemon.

```
systemctl enable ntpd
```

   b. Configure `ntpd` to start when the system starts.

      Currently, an unresolved issue associated with NTP prevents `ntpd` from restarting correctly after a reboot, and the following commands provide a workaround to ensure that it does.

```
echo "systemctl start ntpd" >> /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
```

   c. Start `ntpd`.

```
systemctl start ntpd
```

7. Start Control Center.

```
systemctl start serviced
```

Resolving package dependency conflicts E

This appendix includes procedures for resolving common Docker CE and Control Center dependency conflicts.

Resolving device mapper dependency conflicts

To perform this procedure, you need:

- An RHEL/CentOS system with internet access and the same operating system release and kernel as the Control Center hosts in your deployment.
- A secure network copy program.

Use this procedure to resolve dependency issues in which the installed versions of device mapper libraries are newer than the versions included in the Zenoss mirror. The following example shows a typical `yum` error of this type:

```
Error: Package: 7:device-mapper-event-1.02.107-5.el7.x86_64 (zenoss-
mirror)
Requires: device-mapper = 7:1.02.107-5.el7
Installed: 7:device-mapper-1.02.107-5.el7_2.5.x86_64 (@updates)
device-mapper = 7:1.02.107-5.el7_2.5
```

Follow these steps:

1. Display the version number of the installed device mapper package.

   ```
   rpm -q device-mapper | cut -d - -f 3-
   ```

   Example result:

   ```
   1.02.135-1.el7_3.1.x86_64
   ```

   Record the version number for subsequent use.
2. Log in to a compatible host that is connected to the internet as `root`, or as a user with superuser privileges.

   The host must have the same operating system (RHEL or CentOS) and release installed as the Control Center hosts in your deployment.
3. Install `yum` utilities, if necessary.

a. Determine whether the `yum` utilities package is installed.

```
rpm -qa | grep yum-utils
```

- If the command returns a result, the package is installed. Proceed to the next step.
- If the command does not return a result, the package is not installed. Perform the following substep.

b. Install the `yum-utils` package.

```
yum install yum-utils
```

4. Download the required dependencies, and then create a `tar` archive of the files.
   a. Create a variable for the dependency version to download.
   Replace *Device-Mapper-Version* with the version number displayed in a previous step:

```
myVersion=Device-Mapper-Version
```

   b. Create a temporary directory for the dependencies.

```
mkdir /tmp/downloads
```

   c. Download the dependencies to the temporary directory.

```
yum install --downloadonly --downloaddir=/tmp/downloads \
  device-mapper-event-$myVersion
```

   The `yum` command downloads two package files.
   d. Create a `tar` archive of the temporary directory.

```
cd /tmp && tar czf ./downloads.tgz ./downloads
```

5. Use a secure copy program to copy the archive file to the `/tmp` directory of the Control Center host or hosts that need the dependencies.
6. Log in to the host as `root`, or as a user with superuser privileges.
7. Install the device mapper dependencies.
   a. Extract the packages from the `tar` archive.

```
cd /tmp && tar xzf ./downloads.tgz
```

   b. Install the dependencies.

```
yum install $(ls /tmp/downloads/*.rpm)
```

Return to the procedure you were performing before turning to this appendix and retry the `yum` install command that failed previously.

Resolving other dependency conflicts

Use this procedure to resolve dependency issues in which the installed versions of one or more dependencies are newer than the versions included in the Zenoss mirror. The following example shows a typical `yum` error of this type:

```
Error: Package: policycoreutils-python-2.5-9.el7.x86_64 (zenoss-mirror)
Requires: policycoreutils = 2.5-9.el7
```

```
Installed: policycoreutils-2.5-11.el7_3.x86_64 (@updates)
```

Follow these steps:

1. Install the older package.
   Replace *Package-Name* with the name of the package displayed in the error message:

   ```
   rpm -Uvh --oldpackage Package-Name
   ```

2. Clean all `yum` caches.

   ```
   yum clean all
   ```

Return to the procedure you were performing before turning to this appendix and retry the `yum` install command that failed previously.