Control Center Upgrade Guide

Release 1.2.3

Zenoss, Inc.

www.zenoss.com

Control Center Upgrade Guide

Part Number: 1300.17.058

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

Contents

About this guide

The *Control Center Upgrade Guide* provides detailed instructions for performing the following upgrades:

- Micro upgrades, from version 1.2.x to version 1.2.3
- Minor upgrades, from version 1.1.x to version 1.2.3

Zenoss customers: This guide does not include procedures for upgrading a high-availability deployment. For additional information, please contact your Zenoss representative.

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 a Control Center cluster.
Control Center Installation Guide	Provides detailed procedures for installing and configuring a Control Center cluster.
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.

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 and part number, and as much information as possible about the context of your feedback. The part number appears at the end of the list of trademarks, at the front of this guide.

Supported clients and browsers

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

Client OS	Supported Browsers
Windows 7 and 8.1	Internet Explorer 11 (Enterprise mode only; compatibility mode is not supported.)
	Internet Explorer 10 (Compatibility mode is not supported.)
	Firefox 30 and above
	Chrome 30 and above
Windows Server 2012 R2	Firefox 30
	Chrome 36
Macintosh OS/X 10.9	Firefox 30 and above
	Chrome 36 and above

Client OS	Supported Browsers
Ubuntu 14.04 LTS	Firefox 30 and above
	Chrome 37 and above
Red Hat Enterprise Linux 6.5, CentOS 6.5	Firefox 30 and above
	Chrome 37 and above

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*.

1320.17.058

Update release number (1.2.3).

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

Remove section for configuring `dnsmasq`.

1320.17.024

Update release number (1.2.2).

1300.16.351

Add sections for configuring `dnsmasq` and removing consistent network device naming.

Simplify Docker Engine install and configuration steps.

Add sections about the maintenance scripts that are installed.

1300.16.350

Add a chapter of details about releases and upgrades.

Create a part for the minor release upgrade chapters.

Add a part and new chapters for micro release upgrades.

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

Add a procedure for updating ZooKeeper images on offline nodes (Zenoss customers only).

Clarify the procedures for stopping a cluster.

Update release number (1.2.1).

1300.16.327

Add a description of a new feature, setting the connection timeout value of a resource pool, to the pre-upgrade chapter.

Add a procedure for the new feature to the post-upgrade chapter.

1300.16.322

Initial release (1.2.0).

1 Documented upgrade paths

This chapter includes the dates of Control Center releases and the upgrade paths that are documented in this guide.

Release dates

Table 1: Release 1.2

Release	Date
Control Center 1.2.3	27 Feb 2017
Control Center 1.2.2	25 Jan 2017
Control Center 1.2.1	16 Dec 2016
Control Center 1.2.0	14 Nov 2016

Table 2: Release 1.1

Release	Date
Control Center 1.1.10	23 Nov 2016
Control Center 1.1.9	17 Oct 2016
Control Center 1.1.8	20 Sep 2016
Control Center 1.1.7	20 Jul 2016
Control Center 1.1.6	28 Jun 2016
Control Center 1.1.5	01 Jun 2016
Control Center 1.1.4	24 May 2016
Control Center 1.1.3	20 Apr 2016
Control Center 1.1.2	04 Mar 2016
Control Center 1.1.1	29 Feb 2016

Table 3: Release 1.0

Release	Date
Control Center 1.0.10	20 Feb 2016
Control Center 1.0.9	02 Dec 2015
Control Center 1.0.8	16 Nov 2015
Control Center 1.0.7	10 Oct 2015
Control Center 1.0.6	14 Sep 2015
Control Center 1.0.5	05 Aug 2015
Control Center 1.0.4	10 Jul 2015
Control Center 1.0.3	27 May 2015
Control Center 1.0.2	20 Apr 2015
Control Center 1.0.1	03 Apr 2015
Control Center 1.0.0	24 Feb 2015

Control Center releases and image tags

The table in this section associates Control Center releases with the tags of the required Docker Engine images for each release. The images provide the virtual containers of the Control Center internal services and the ZooKeeper service.

Release 1.2

Release	Internal services image tag	ZooKeeper image tag
Control Center 1.2.3	zenoss/serviced-isvcs:v??	zenoss/isvcs-zookeeper:v?
Control Center 1.2.2	zenoss/serviced-isvcs:v??	zenoss/isvcs-zookeeper:v?
Control Center 1.2.1	zenoss/serviced-isvcs:v54	zenoss/isvcs-zookeeper:v8
Control Center 1.2.0	zenoss/serviced-isvcs:v53	zenoss/isvcs-zookeeper:v8

Upgrade paths included in this document

Micro release upgrade paths

From	To
Control Center 1.2.2	Control Center 1.2.3
Control Center 1.2.1	Control Center 1.2.3
Control Center 1.2.0	Control Center 1.2.3

Minor release upgrade paths

From	To
Control Center 1.1.10	Control Center 1.2.3
Control Center 1.1.9	Control Center 1.2.3
Control Center 1.1.8	Control Center 1.2.3
Control Center 1.1.7	Control Center 1.2.3
Control Center 1.1.5	Control Center 1.2.3
Control Center 1.1.4	Control Center 1.2.3
Control Center 1.1.3	Control Center 1.2.3
Control Center 1.1.2	Control Center 1.2.3
Control Center 1.1.1	Control Center 1.2.3

Upgrading 1.2.x to 1.2.3

Scope

The chapters in this part provide instructions for performing a micro upgrade of Control Center, from version 1.2.x to 1.2.3.

The following table identifies the upgrades of Control Center that are documented in this part.

From	To
Control Center 1.2.2	Control Center 1.2.3
Control Center 1.2.1	Control Center 1.2.3
Control Center 1.2.0	Control Center 1.2.3

The following list outlines recommended best practices for upgrading Control Center deployments:

1. Download a copy of the *Control Center Release Notes* for this release and review its contents. The latest information is included in that document.
2. On delegate hosts, most of the upgrade steps are identical. Use `tmux` or a similar program to establish sessions on each delegate host and perform the steps at the same time.
3. Review and verify the settings in delegate host configuration files (`/etc/default/serviced`) before starting the upgrade. Ideally, the settings on all delegate hosts are identical, except on ZooKeeper nodes and delegate hosts that do not mount the DFS.
4. Review the procedures in this guide before performing them. Every effort is made to avoid mistakes and anticipate needs; nevertheless, the instructions may be incorrect or inadequate for some requirements or environments.
5. **Zenoss customers:** Download and stage files for offline upgrades, if necessary. For more information, see *Preparing to install without internet access* on page 74.

2 Stopping a Control Center deployment

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

Note The procedures in this chapter assume that Control Center is the only source of Docker Engine containers that are run on Control Center cluster hosts.

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
```

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 (multi-host deployment)

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 Engine services.

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

d Start the NFS and Docker Engine 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
```

- 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)
do
  umount -f $FS
done
```

c Identify filesystems mounted from the master host.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts
```

- 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)
do
  umount -f -l $FS
done
```

e Restart the NFS service.

```
systemctl restart nfs
```

f Determine whether any filesystems remain mounted.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts
```

- 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
```

Upgrading 1.2.x to 1.2.3

3

Perform the procedures in this chapter to upgrade a Control Center 1.2.x system to 1.2.3.

Note **Zenoss customers**: If you are upgrading a deployment that does not have internet access, perform the steps in *Preparing to install without internet access* on page 74 before performing the procedures in this chapter.

Updating Control Center on the master host

Use this procedure to update Control Center on the master host to version 1.2.3.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Save the current `serviced` configuration file as a reference.
   a. Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.2.3
```

   b. Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.2.3
```

3. Install the new version of Control Center.
   With internet access:

```
yum clean all && yum makecache fast
yum --enablerepo=zenoss-stable install -y serviced-1.2.3
```

   Without internet access:

```
yum clean all && yum makecache fast
yum install -y --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.2.3-1.x86_64.rpm
```

4. Make a backup copy of the new configuration file.
   a. Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.2.3-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.2.3-orig
```

5 Compare the new configuration file with the configuration file of the previous release.

a Identify the configuration files to compare.

```
ls -l /etc/default/serviced*
```

The original versions of the configuration files should end with `orig`, but you may have to compare the dates of the files.

b Compare the new and previous configuration files.
Replace *New-Version* with the name of the new configuration file, and replace *Previous-Version* with the name of the previous configuration file:

```
diff New-Version Previous-Version
```

For example, to compare versions 1.2.0 and 1.2.3, enter the following command:

```
diff /etc/default/serviced-1.2.0-orig \
  /etc/default/serviced-1.2.3-orig
```

- If the command returns no result, restore the backup of the previous configuration file.

```
cp /etc/default/serviced-pre-1.2.3 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

- If the command returns a result, restore the backup of the previous configuration file, and then use the results to edit the restored version, if necessary or desired.

```
cp /etc/default/serviced-pre-1.2.3 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

For more information about configuring the master host, see *Control Center configuration variables* on page 52.

- If you are upgrading a multi-host deployment, continue to the next procedure.
- If you are upgrading a single-host deployment, start Control Center.

```
systemctl start serviced
```

Updating Control Center on delegate hosts

This procedure updates Control Center on delegate hosts to version 1.2.3.

Perform this procedure on each delegate host in your deployment.

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

2 Save the current `serviced` configuration file as a reference.

a Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.2.3
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.2.3
```

3 Install the new version of Control Center.
With internet access:

```
yum clean all && yum makecache fast
yum --enablerepo=zenoss-stable install -y serviced-1.2.3
```

Without internet access:

```
yum clean all && yum makecache fast
yum install -y --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.2.3-1.x86_64.rpm
```

4 Make a backup copy of the new configuration file.

a Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.2.3-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.2.3-orig
```

5 Compare the new configuration file with the configuration file of the previous release.

a Identify the configuration files to compare.

```
ls -l /etc/default/serviced*
```

The original versions of the configuration files should end with `orig`, but you may have to compare the dates of the files.

b Compare the new and previous configuration files.
Replace *New-Version* with the name of the new configuration file, and replace *Previous-Version* with the name of the previous configuration file:

```
diff New-Version Previous-Version
```

For example, to compare versions 1.2.0 and 1.2.3, enter the following command:

```
diff /etc/default/serviced-1.2.0-orig \
  /etc/default/serviced-1.2.3-orig
```

- If the command returns no result, restore the backup of the previous configuration file.

```
cp /etc/default/serviced-pre-1.2.3 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

- If the command returns a result, restore the backup of the previous configuration file, and then use the results to edit the restored version, if necessary or desired.

```
cp /etc/default/serviced-pre-1.2.3 \
  /etc/default/serviced && chmod 0644 /etc/default/serviced
```

For more information about configuring a delegate host, see *Control Center configuration variables* on page 52.

Updating the ZooKeeper image on ensemble nodes that do not have internet access

Before performing this procedure, review the information in the following table to determine whether it is necessary.

Release	Internal services image tag	ZooKeeper image tag
Control Center 1.2.3	zenoss/serviced-isvcs:v??	zenoss/isvcs-zookeeper:v?
Control Center 1.2.2	zenoss/serviced-isvcs:v??	zenoss/isvcs-zookeeper:v?
Control Center 1.2.1	zenoss/serviced-isvcs:v54	zenoss/isvcs-zookeeper:v8
Control Center 1.2.0	zenoss/serviced-isvcs:v53	zenoss/isvcs-zookeeper:v8

- If the tags of the ZooKeeper images are the same for the micro upgrade you are performing, skip this procedure.
- If the tags of the ZooKeeper images are not the same, and you are a Zenoss customer, perform this procedure.

Zenoss customers: Perform the steps in *Preparing to install without internet access* on page 74 before performing this procedure.

Use this procedure to install a new Docker Engine image for ZooKeeper on ZooKeeper ensemble nodes that do not have internet access. Control Center pulls the required image from Docker Hub on ZooKeeper ensemble nodes that have internet access.

Perform the following steps:

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Identify the hosts in the ZooKeeper ensemble.

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

   The result is a list of 3 or 5 hosts, separated by the comma character (`,`). The master host is always a node in the ZooKeeper ensemble.
3. Log in to a ZooKeeper ensemble node as `root`, or as a user with superuser privileges.
4. Change directory to `/root`.

   ```
   cd /root
   ```
5. Extract the ZooKeeper image.

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

   Image extraction begins when you press the **y** key.
6. Optional: Delete the archive file, if desired.

   ```
   rm -i ./install-zenoss-isvcs-zookeeper:v*.run
   ```
7. Repeat the preceding four steps on each ZooKeeper node in the ensemble.

Starting the ZooKeeper ensemble

Use this procedure to start the ZooKeeper ensemble.

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 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.

```
serviced service attach serviced-isvcs_zookeeper
```

   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`.
   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.

Upgrading 1.1.x to 1.2.3
Scope

The chapters in this part provide instructions for performing a minor upgrade of Control Center, from version 1.1.x to 1.2.3.

The following table identifies the upgrades of Control Center that are documented in this part.

From	To
Control Center 1.1.10	Control Center 1.2.3
Control Center 1.1.9	Control Center 1.2.3
Control Center 1.1.8	Control Center 1.2.3
Control Center 1.1.7	Control Center 1.2.3
Control Center 1.1.5	Control Center 1.2.3
Control Center 1.1.4	Control Center 1.2.3
Control Center 1.1.3	Control Center 1.2.3
Control Center 1.1.2	Control Center 1.2.3
Control Center 1.1.1	Control Center 1.2.3

4 Before upgrading

This chapter provides information and procedures to prepare your Control Center deployment for an upgrade.

New features that affect upgrades

This release includes new features and new requirements that affect the upgrade process. The following list provides an overview of the changes that are addressed during this upgrade:

- The minimum kernel version for Control Center hosts is `3.10.0-327.22.2.el7.x86_64`. The kernel is available for RHEL/CentOS 7.1 and 7.2. For optimal results, the 7.2 release is recommended, and the instructions for checking and updating the kernel include an option to update the operating system at the same time. To prevent dependency issues, updating the kernel or operating system is a step in the Docker Engine update procedure.
- The `serviced` configuration file includes many new variables since version 1.1.1, and some deprecated variables. For more information, see *Control Center configuration variables* on page 52.
- All delegate communications are authenticated. To enable this feature, all existing hosts must install unique credentials, which are generated on the master host. The installation steps are included in the startup procedures.
- With delegate authentication, Control Center can control administrative and DFS access permissions at the resource pool level. During the upgrade, all existing resource pools are given both administrative and DFS access permissions. The post-upgrade chapter includes an optional procedure for removing permissions from a resource pool.
- In previous releases, the *SERVICED_NFS_CLIENT* variable was set on delegate hosts to prevent access to the DFS. In this release, *SERVICED_NFS_CLIENT* is deprecated in favor of setting DFS access permission at the resource pool level. To ease the transition to the new functionality, delegate host configurations that include the *SERVICED_NFS_CLIENT* variable are still supported.
- This release includes a new resource pool feature, the ability to set the length of time the scheduler waits for a disconnected delegate host to rejoin its pool before moving the services scheduled for the delegate to a different host in the pool. This feature is useful for remote resource pools that are connected through a high-latency, wide-area network.
- Among other changes since version 1.9.0, Docker Engine 1.12.1 includes a new storage subsystem. The initial startup takes a little longer, as the old layout is replaced with the new layout.

For more information about this release, refer to the *Control Center Release Notes*.

Upgrade best practices

The following list outlines recommended best practices for upgrading Control Center deployments:

1. Download a copy of the *Control Center Release Notes* for this release and review its contents. The latest information is included in that document.
2. On delegate hosts, most of the upgrade steps are identical. Use `tmux` or a similar program to establish sessions on each delegate host and perform the steps at the same time.
3. Review and verify the settings in delegate host configuration files (`/etc/default/serviced`) before starting the upgrade. Ideally, the settings on all delegate hosts are identical, except on ZooKeeper nodes and delegate hosts that do not mount the DFS.
4. Review the procedures in this guide before performing them. Every effort is made to avoid mistakes and anticipate needs; nevertheless, the instructions may be incorrect or inadequate for some requirements or environments.
5. **Zenoss customers:** Download and stage files for offline upgrades, if necessary. For more information, see *Preparing to install without internet access* on page 74.

5 Stopping a Control Center deployment

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

Note The procedures in this chapter assume that Control Center is the only source of Docker Engine containers that are run on Control Center cluster hosts.

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
```

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 (multi-host deployment)

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 Engine services.

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

d Start the NFS and Docker Engine 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
```

- 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)
do
  umount -f $FS
done
```

c Identify filesystems mounted from the master host.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts
```

- 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)
do
  umount -f -l $FS
done
```

e Restart the NFS service.

```
systemctl restart nfs
```

f Determine whether any filesystems remain mounted.

```
awk '/serviced/ { print $1, $2 }' < /proc/mounts
```

- 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
```

6 Upgrading the Control Center master host

Perform the procedures in this chapter to upgrade the master host.

Note **Zenoss customers**: If you are upgrading your deployment without internet access, perform the steps in *Preparing to install without internet access* on page 74 before performing the procedures in this chapter.

Disabling consistent network device naming

The consistent network device naming feature can create device names that are too long for use with virtual IPs. This step disables the feature so that all network device names are `eth` followed by an integer.

1. Log in as `root`, or as a user with superuser privileges.
2. Edit the GRUB 2 general settings file.
   a. Create a backup copy of the settings file.

      ```
      cp /etc/default/grub /etc/default/grub.bak
      ```

   b. Open `/etc/default/grub` with a text editor.
   c. Add kernel boot arguments to the value of the *GRUB_CMDLINE_LINUX* variable.

      The arguments to add are `biosdevname=0` and `net.ifnames=0`. Make sure the arguments are between the delimiter characters (`"`).
   d. Save the file, and then close the text editor.
3. Determine whether the host is configured for UEFI or legacy boot mode.

   ```
   find /boot -maxdepth 1 -type d
   ```

   - If the output includes `/boot/efi`, the host is configured for UEFI boot mode.
   - If the output does not include `/boot/efi`, the host is configured for legacy boot mode.
4. Recreate the GRUB 2 boot configuration file.
   For UEFI boot mode:

   ```
   grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
   ```

   For legacy boot mode:

   ```
   grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

5 Create an override for the default rules policy of the dynamic device management daemon.

```
ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules
```

6 Change network device configurations.

a Change directory to `/etc/sysconfig/network-scripts`.

```
cd /etc/sysconfig/network-scripts
```

b List the network device configuration files.

```
ls ifcfg-* | grep -v ifcfg-lo
```

The `ifcfg-lo` file is excluded because it does not need to be changed.

c Rename and edit device files.

For example, if you have a device file named `ifcfg-eno16777736`, use the `mv` command rename it to `ifcfg-eth0`, and then change the values of the *NAME* and *DEVICE* variables inside the file to `eth0`.

7 Reboot the host.

Updating Docker Engine

Use this procedure to update Docker Engine to version 1.12.1.

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

2 Ensure that Docker Engine updates are disabled.

During a kernel update, `yum` stops processing when it finds the `serviced` dependency on Docker Engine 1.9.0. Docker Engine updates are explicitly enabled and then disabled in subsequent steps.

a Check the Docker repository file.

```
grep enabled /etc/yum.repos.d/docker.repo
```

- If the result is `enabled=0`, proceed to the next step.
- If the result is `enabled=1`, perform the following substeps.

b Open `/etc/yum.repos.d/docker.repo` with a text editor.

c Change the value of the `enabled` key from `1` to `0`.

d Save the file, and then close the text editor.

3 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 substeps.

b Disable automatic start of `serviced`.

```
systemctl disable serviced
```

c 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 -y kernel && reboot
```

d. Log in as `root`, or as a user with superuser privileges.
e. Enable automatic start of `serviced`.

```
systemctl enable serviced
```

4. Enable Docker Engine updates.
   a. Open `/etc/yum.repos.d/docker.repo` with a text editor.
   b. Change the value of the `enabled` key from `0` to `1`.
   c. Save the file, and then close the text editor.
5. Identify the name of the LVM thin pool for Docker Engine.

```
docker info 2>/dev/null | grep 'Pool Name'
```

Example result:

```
 Pool Name: docker-docker--pool
```

Record the name for use in a subsequent step.
6. Back up the Docker Engine environment file.

```
test -f /etc/sysconfig/docker \
  && mv /etc/sysconfig/docker /etc/sysconfig/docker.bak
```

7. Stop the Docker Engine service.

```
systemctl stop docker
```

8. Remove Docker Engine 1.9.0 without checking dependencies.

```
rpm -e --nodeps docker-engine-1.9.0
```

9. Install Docker Engine 1.12.1.
With internet access:

```
yum clean all && yum makecache fast
yum install -y docker-engine-1.12.1
```

Without internet access:

```
yum clean all && yum makecache fast
yum install --enablerepo=zenoss-mirror -y docker-engine-1.12.1
```

10. Disable unintended Docker Engine updates.
    a. Open `/etc/yum.repos.d/docker.repo` with a text editor.
    b. Change the value of the `enabled` key from `1` to `0`.
    c. Save the file, and then close the text editor.

Configuring Docker Engine

Use this procedure to configure Docker Engine.

1. Log in as `root`, or as a user with superuser privileges.
2. Create a new Docker Engine drop-in file.

a Create a directory for the drop-in file, if necessary.

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

b Create the 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
```

3 Reload the `systemd` manager configuration.

```
systemctl daemon-reload
```

4 Configure and start the Docker Engine service.

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

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

```
myPool="/dev/mapper/Thin-Pool-Device"
```

b Create a variable for the master host.

Replace *Master-Host* with the hostname or IPv4 address of the master host:

```
masterHost="Master-Host"
```

The value of this variable must match the value of the *SERVICED_DOCKER_REGISTRY* variable in `/etc/default/serviced`.

c Create variables for adding arguments to the Docker Engine environment 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=$myPool"
myReg="--insecure-registry=$masterHost:5000"
```

d Add the arguments to the Docker Engine environment file.

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

e Start or restart Docker.

```
systemctl restart docker
```

The startup may take up to a minute as Docker Engine updates the storage layout.

5 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 Restart the Docker service.

```
systemctl restart docker
```

6 Compare the previous version of the Docker Engine environment file with the new version, and add any customizations for your deployment to the new version.

```
diff /etc/sysconfig/docker /etc/sysconfig/docker.bak
```

If you change this file, restart Docker Engine with `systemctl restart docker`.

Optional: Loading image files

Zenoss customers: Perform the steps in *Preparing to install without internet access* on page 74 before performing this procedure.

Use this procedure to load images into the local repository on a host that does not have internet access.

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

2 Change directory to `/root`.

```
cd /root
```

3 Extract the images.

```
for image in install-*.run
do
echo -n "$image: "
eval ./$image
done
```

Image extraction begins when you press the **y** key. If you press the **y** key and then **Return** key, the current image is extracted, but the next one is not.

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-*.run
```

Updating Control Center on the master host

Use this procedure to update Control Center on the master host to version 1.2.3.

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

2 Save the current `serviced` configuration file as a reference.

a Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.2.3
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.2.3
```

3 Install the new version of Control Center.

With internet access:

```
yum clean all && yum makecache fast
yum --enablerepo=zenoss-stable install -y serviced-1.2.3
```

Without internet access:

```
yum clean all && yum makecache fast
yum install -y --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.2.3-1.x86_64.rpm
```

4 Make a backup copy of the new configuration file.

a Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.2.3-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.2.3-orig
```

5 Display the settings of the reference configuration file.

```
grep -E '^\b*SERVICED' /etc/default/serviced-pre-1.2.3
```

6 Open the new configuration file with a text editor, and then update the file for your environment.

For more information about configuring the master host, see *Control Center configuration variables* on page 52.

7 Upgrading Control Center delegate hosts

Perform the procedures in this chapter to upgrade delegate hosts.

Note **Zenoss customers**: If you are upgrading your deployment without internet access, perform the steps in *Preparing to install without internet access* on page 74 before performing the procedures in this chapter.

Disabling consistent network device naming

The consistent network device naming feature can create device names that are too long for use with virtual IPs. This step disables the feature so that all network device names are `eth` followed by an integer.

1. Log in as `root`, or as a user with superuser privileges.
2. Edit the GRUB 2 general settings file.
   a. Create a backup copy of the settings file.

      ```
      cp /etc/default/grub /etc/default/grub.bak
      ```

   b. Open `/etc/default/grub` with a text editor.
   c. Add kernel boot arguments to the value of the *GRUB_CMDLINE_LINUX* variable.

      The arguments to add are `biosdevname=0` and `net.ifnames=0`. Make sure the arguments are between the delimiter characters (`"`).
   d. Save the file, and then close the text editor.
3. Determine whether the host is configured for UEFI or legacy boot mode.

   ```
   find /boot -maxdepth 1 -type d
   ```

   - If the output includes `/boot/efi`, the host is configured for UEFI boot mode.
   - If the output does not include `/boot/efi`, the host is configured for legacy boot mode.
4. Recreate the GRUB 2 boot configuration file.
   For UEFI boot mode:

   ```
   grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
   ```

   For legacy boot mode:

   ```
   grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

5 Create an override for the default rules policy of the dynamic device management daemon.

```
ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules
```

6 Change network device configurations.

a Change directory to `/etc/sysconfig/network-scripts`.

```
cd /etc/sysconfig/network-scripts
```

b List the network device configuration files.

```
ls ifcfg-* | grep -v ifcfg-lo
```

The `ifcfg-lo` file is excluded because it does not need to be changed.

c Rename and edit device files.

For example, if you have a device file named `ifcfg-eno16777736`, use the `mv` command rename it to `ifcfg-eth0`, and then change the values of the *NAME* and *DEVICE* variables inside the file to `eth0`.

7 Reboot the host.

Updating Docker Engine

Use this procedure to update Docker Engine to version 1.12.1.

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

2 Ensure that Docker Engine updates are disabled.

During a kernel update, `yum` stops processing when it finds the `serviced` dependency on Docker Engine 1.9.0. Docker Engine updates are explicitly enabled and then disabled in subsequent steps.

a Check the Docker repository file.

```
grep enabled /etc/yum.repos.d/docker.repo
```

- If the result is `enabled=0`, proceed to the next step.
- If the result is `enabled=1`, perform the following substeps.

b Open `/etc/yum.repos.d/docker.repo` with a text editor.

c Change the value of the `enabled` key from `1` to `0`.

d Save the file, and then close the text editor.

3 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 substeps.

b Disable automatic start of `serviced`.

```
systemctl disable serviced
```

c 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 -y kernel && reboot
```

d Log in as `root`, or as a user with superuser privileges.

e Enable automatic start of `serviced`.

```
systemctl enable serviced
```

4 Enable Docker Engine updates.

a Open `/etc/yum.repos.d/docker.repo` with a text editor.

b Change the value of the `enabled` key from `0` to `1`.

c Save the file, and then close the text editor.

5 Identify the name of the LVM thin pool for Docker Engine.

```
docker info 2>/dev/null | grep 'Pool Name'
```

Example result:

```
 Pool Name: docker-docker--pool
```

Record the name for use in a subsequent step.

6 Back up the Docker Engine environment file.

```
test -f /etc/sysconfig/docker \
  && mv /etc/sysconfig/docker /etc/sysconfig/docker.bak
```

7 Stop the Docker Engine service.

```
systemctl stop docker
```

8 Remove Docker Engine 1.9.0 without checking dependencies.

```
rpm -e --nodeps docker-engine-1.9.0
```

9 Install Docker Engine 1.12.1.
With internet access:

```
yum clean all && yum makecache fast
yum install -y docker-engine-1.12.1
```

Without internet access:

```
yum clean all && yum makecache fast
yum install --enablerepo=zenoss-mirror -y docker-engine-1.12.1
```

10 Disable unintended Docker Engine updates.

a Open `/etc/yum.repos.d/docker.repo` with a text editor.

b Change the value of the `enabled` key from `1` to `0`.

c Save the file, and then close the text editor.

Configuring Docker Engine

Use this procedure to configure Docker Engine.

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

2 Create a new Docker Engine drop-in file.

a Create a directory for the drop-in file, if necessary.

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

b Create the 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
```

3 Reload the `systemd` manager configuration.

```
systemctl daemon-reload
```

4 Configure and start the Docker Engine service.

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

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

```
myPool="/dev/mapper/Thin-Pool-Device"
```

b Create a variable for the master host.

Replace *Master-Host* with the hostname or IPv4 address of the master host:

```
masterHost="Master-Host"
```

The value of this variable must match the value of the *SERVICED_DOCKER_REGISTRY* variable in `/etc/default/serviced`.

c Create variables for adding arguments to the Docker Engine environment 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=$myPool"
myReg="--insecure-registry=$masterHost:5000"
```

d Add the arguments to the Docker Engine environment file.

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

e Start or restart Docker.

```
systemctl restart docker
```

The startup may take up to a minute as Docker Engine updates the storage layout.

5 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 Restart the Docker service.

```
systemctl restart docker
```

6 Compare the previous version of the Docker Engine environment file with the new version, and add any customizations for your deployment to the new version.

```
diff /etc/sysconfig/docker /etc/sysconfig/docker.bak
```

If you change this file, restart Docker Engine with `systemctl restart docker`.

Updating Control Center on delegate hosts

This procedure updates Control Center on delegate hosts to version 1.2.3.

Perform this procedure on each delegate host in your deployment.

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

2 Save the current `serviced` configuration file as a reference.

a Rename the file.

```
mv /etc/default/serviced /etc/default/serviced-pre-1.2.3
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-pre-1.2.3
```

3 Install the new version of Control Center.
With internet access:

```
yum clean all && yum makecache fast
yum --enablerepo=zenoss-stable install -y serviced-1.2.3
```

Without internet access:

```
yum clean all && yum makecache fast
yum install -y --enablerepo=zenoss-mirror \
  /opt/zenoss-repo-mirror/serviced-1.2.3-1.x86_64.rpm
```

4 Make a backup copy of the new configuration file.

a Copy the file.

```
cp /etc/default/serviced /etc/default/serviced-1.2.3-orig
```

b Set permissions to read-only.

```
chmod 0440 /etc/default/serviced-1.2.3-orig
```

5 Display the settings of the reference configuration file.

```
grep -E '^\b*SERVICED' /etc/default/serviced-pre-1.2.3
```

6 Open the new configuration file with a text editor, and then update the file for your environment.

Among other changes, the *SERVICED_NFS_CLIENT* variable is deprecated. Make note of the delegates that use the setting, and then rescind DFS access permission from the resource pool to which they belong. For more information, see *Updating resource pool permissions* on page 47.

For more information about configuring a delegate host, see *Control Center configuration variables* on page 52.

Updating the ZooKeeper image on ensemble nodes that do not have internet access

Zenoss customers: Perform the steps in *Preparing to install without internet access* on page 74 before performing this procedure.

Use this procedure to install a new Docker Engine image for ZooKeeper on ZooKeeper ensemble nodes that do not have internet access. Control Center pulls the required image from Docker Hub on ZooKeeper ensemble nodes that have internet access.

Perform the following steps:

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

2 Identify the hosts in the ZooKeeper ensemble.

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

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

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

4 Change directory to `/root`.

```
cd /root
```

5 Extract the ZooKeeper image.

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

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

6 Optional: Delete the archive file, if desired.

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

7 Repeat the preceding four steps on each ZooKeeper node in the ensemble.

8 Starting an upgraded deployment

This chapter includes procedures for starting single-host and multi-host Control Center deployments after upgrading to version 1.2.3.

Single-host deployments: *Starting and registering the master host* on page 42.

Multi-host deployments: Use the procedures in this section in the order shown in the following table.

Step		Procedure
1	Start/register the master host	*Starting and registering the master host* on page 42
2	Start/register ZooKeeper ensemble hosts	*Updating delegate hosts with authentication* on page 43
3	Start the ZooKeeper ensemble	*Starting the ZooKeeper ensemble* on page 20
4	Start/register all remaining delegate hosts	*Updating delegate hosts with authentication* on page 43

Starting and registering the master host

Use this procedure to start the master host after upgrading Control Center to version 1.2.3. This procedure also includes steps to create and register the authentication credentials the master needs for its role as a delegate.

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

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

4. Start `serviced`, and then monitor the startup.

During this startup, `serviced` invokes `docker pull` to retrieve its updated images.

```
systemctl start serviced && journalctl -u serviced -f -o cat
```

Do not proceed to the next step until the following message is displayed:

```
Host Master successfully started
```

5 Obtain the host ID of the master host.

a Display the host IDs of all cluster hosts.

```
serviced host list | cut -c-85
```

b Record the host ID of the master host.

6 Create authentication credentials for the master host, and register the credentials.
Replace *Host-ID* with the host ID of the master host:

```
serviced key reset --register Host-ID
```

7 Optional: Identify the hosts in the ZooKeeper ensemble.

This step is only required for multi-host deployments.

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

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

Updating delegate hosts with authentication

Control Center requires authentication tokens for all delegate communications. The tokens are based on RSA key pairs created by the master `serviced` instance. When you create a key pair for a delegate, `serviced` bundles its public key with the delegate's private key. The `serviced` instance on the delegate installs the credentials and uses them to sign messages with the required unique tokens.

Credentials are installed by using an SSH connection or a file.

- The command to create a key pair can initiate an SSH connection with a delegate and install credentials. 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 create a key pair creates a file containing the credentials. You can move the credentials file to the delegate host with any file transfer method, and then install it on the delegate.

The following procedures demonstrate how to create credentials and install them on a delegate.

Starting and registering a delegate using SSH

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 start a delegate host after upgrading Control Center to version 1.2.3. This procedure also includes steps to create and register the authentication credentials the delegate needs, through an SSH connection.

1. Log in to the delegate 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. Verify the settings in the `serviced` configuration file.

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

4. Start `serviced`, and then monitor the startup.

   During this startup, `serviced` invokes `docker pull` to retrieve its updated images.

   ```
   systemctl start serviced && journalctl -u serviced -f -o cat
   ```

   Do not proceed to the next step until the following message is displayed:

   ```
   Host Agent successfully started
   ```

5. Log out of the delegate host.
6. Log in to the master host as `root`, or as a user with superuser privileges.
7. Obtain the host ID of the delegate host started previously.
   a. Display the host IDs of all cluster hosts.

      ```
      serviced host list | cut -c-85
      ```

   b. Record the host ID of the delegate host.
8. Create authentication credentials for the delegate host, and register the credentials.

   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.

   Replace *Host-ID* with the host ID of the delegate host started previously:

   ```
   serviced key reset --register Host-ID
   ```

Starting and registering a delegate using a file

Use this procedure to start a delegate host after upgrading Control Center to version 1.2.3. This procedure also includes steps to create a credentials file and to use the file to register the delegate.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Obtain the host ID of the delegate host to start and register.
   a. Display the host IDs of all cluster hosts.

      ```
      serviced host list | cut -c-85
      ```

b Record the host ID of the delegate host.

3 Create authentication credentials for the delegate host.
Replace *Host-ID* with the host ID of the delegate host identified in the preceding step:

```
serviced key reset Host-ID
```

The command creates a unique credentials file in the local directory.

4 Use a file transfer utility such as `scp` to copy the credentials file to the delegate host.
Once copied to the delegate host, the credentials file is not needed on the master host and can be deleted.

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

6 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
```

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

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

8 Start `serviced`, and then monitor the startup.
During this startup, `serviced` invokes `docker pull` to retrieve its updated images.

```
systemctl start serviced && journalctl -u serviced -f -o cat
```

Do not proceed to the next step until the following message is displayed:

```
Host Agent successfully started
```

9 Install the credentials.
Replace *Credentials-File* with the pathname of the credentials file:

```
serviced host register Credentials-File
```

10 Delete the credentials file.
The file is no longer needed on the delegate host.
Replace *Credentials-File* with the pathname of the credentials file:

```
rm Credentials-File
```

Starting the ZooKeeper ensemble

Use this procedure to start the ZooKeeper ensemble.

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 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.

```
serviced service attach serviced-isvcs_zookeeper
```

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`.

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.

9 After upgrading

Perform the procedures in this chapter after Control Center is upgraded.

Updating resource pool permissions

Use this procedure to identify and change the permissions associated with one or more resource pools. During the upgrade to version 1.2.3, existing resource pools are assigned both administrative and DFS permissions.

In this release, the command that displays information about pools uses integer values for the Permissions field. The following list associates the values with the permissions they represent:

`1`, administrative permission
`2`, DFS access permission
`3`, both administrative and DFS access permissions

1. Log in to the master host as a user with `serviced` CLI privileges.
2. Display the list of resource pools and their permissions.

   ```
   serviced pool list -v | grep -E 'ID|Permissions'
   ```

   Example result:

   ```
   "ID": "default",
   "Permissions": 3,
   "ID": "master",
   "Permissions": 3,
   ```

   In the preceding example, the `default` and `master` resource pools have administrative permission and DFS access permission.
3. Optional: Remove DFS access permission from a pool, if desired.

   If you intend to remove both DFS access and administrative access permissions from a resource pool, you must remove DFS access permissions first.

   Replace *Pool-Name* with the name of a resource pool from the previous step:

   ```
   serviced pool set-permission --dfs=false Pool-Name
   ```
4. Optional: Remove administrative permission from a pool, if desired.

If you intend to remove both DFS access and administrative access permissions from a resource pool, you must remove DFS access permissions first.

Replace *Pool-Name* with the name of a resource pool from a previous step:

```
serviced pool set-permission --admin=false Pool-Name
```

Setting the connection timeout of a resource pool

Use this procedure to set the length of time the scheduler waits for a disconnected delegate host to rejoin its pool before moving the services scheduled for the delegate to a different host in the pool. This feature is useful for remote resource pools that are connected through a high-latency, wide-area network.

1. Log in to the master host as a user with `serviced` CLI privileges.
2. Display the list of resource pools and their connection timeout values.

```
serviced pool list -v | grep -E 'ID|ConnectionTimeout'
```

3. Optional: Set the connection timeout value of a resource pool, if desired.

   This command accepts the following units identifiers:

   `ms` (milliseconds)
   `s` (seconds)
   `m` (minutes)
   `h` (hours)

   Replace *Pool-ID* with a resource pool identifier, and replace *Timeout+Units* with an integer followed by a units identifier:

```
serviced pool set-conn-timeout Pool-ID Timeout+Units
```

Removing unused images

Use this procedure to identify and remove unused Control Center images.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Identify the images associated with the installed version of `serviced`.

```
serviced version | grep Images
```

   Example result:

```
IsvcsImages: [zenoss/serviced-isvcs:v50 zenoss/isvcs-zookeeper:v7]
```

3. Display the `serviced` images in the local repository.

```
docker images | awk '/REPO|isvcs/'
```

   Example result (edited to fit):

```
REPOSITORY                      TAG       IMAGE ID
zenoss/serviced-isvcs           v40       88cd6c24cc82
zenoss/serviced-isvcs           v50       0aab5a2123f2
zenoss/isvcs-zookeeper          v3        46fa0a2fc4bf
```

```
zenoss/isvcs-zookeeper          v7          0ff3b3117fb8
```

The example result shows the current versions and one set of previous versions. Your result may include additional previous versions and will show different images IDs.

4 Remove unused images.
Replace *Image-ID* with the image ID of an image for a previous version.

```
docker rmi Image-ID
```

Repeat this command for each unused image.

Removing orphaned snapshot devices

Use this procedure to identify orphaned snapshot devices in the LVM thin pool for application data, and to remove them.

1 Log in to the master host as `root`, or as a user with superuser privileges.
2 Identify the location of tenant volumes.

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

- If the command returns a result, the location of tenant volumes is the value of the *SERVICED_VOLUMES_PATH* variable.
- If the command does not return a result, the location of tenant volumes is the default value of *SERVICED_VOLUMES_PATH*, `/opt/serviced/var/volumes`.

3 Identify the device of the `serviced` thin pool belongs.

```
ls /dev/mapper | grep serviced
```

Example result:

```
serviced-serviced--pool
serviced-serviced--pool_tdata
serviced-serviced--pool_tmeta
```

The first result is the thin pool device. The other results represent the data and metadata portions of the device.

4 Check for orphaned snapshot devices.
Replace *Thin-Pool-Device* with the name of the thin pool device from the previous step, and replace *Volumes-Path* with the location of tenant volumes:

```
serviced-storage -o dm.thinpooldev=/dev/mapper/Thin-Pool-Device \
  check Volumes-Path
```

- If the result is `No orphaned devices found`, stop. There are no orphaned snapshot devices to remove.
- If the result is `Orphaned devices were found`, perform the next step.

5 Remove orphaned snapshot devices.
Replace *Thin-Pool-Device* with the name of the thin pool device from the previous step, and replace *Volumes-Path* with the location of tenant volumes:

```
serviced-storage -o dm.thinpooldev=/dev/mapper/Thin-Pool-Device \
```

```
check -c Volumes-Path
```

Updating the OpenTSDB time-to-live value

Older versions of Control Center used a longer time-to-live (TTL) value for data maintained by the OpenTSDB database. The correct value for this version is 2592000 seconds (30 days). Use this procedure to update the TTL value, if necessary.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Start an interactive shell in the OpenTSDB container.

```
docker exec -it serviced-isvcs_opentsdb bash
```

3. Start an interactive HBase shell.

```
/opt/hbase/bin/hbase shell
```

Example result:

```
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.94.16, r1557241, Fri Jan 10 20:43:03 UTC 2014

hbase(main):001:0>
```

4. Display the current settings of the `tsdb` table.

```
describe 'tsdb'
```

   - If the result includes `TTL => '2592000'`, stop. Use the `exit` command twice, to end the HBase shell and then the shell in the OpenTSDB container.
   - If the result includes a larger value for the TTL setting, perform the remaining steps.

5. Disable the `tsdb` table.

```
disable 'tsdb'
```

6. Set the TTL value to 2592000 seconds (30 days).

```
alter 'tsdb', {NAME=>'t', TTL=>'2592000'}
```

7. Enable the `tsdb` table.

```
enable 'tsdb'
```

8. Display the current settings.

```
describe 'tsdb'
```

   - If the result includes `TTL => '2592000'`, proceed to the next step.
   - If the result includes a different value for the TTL setting, repeat the preceding steps.

9. End the interactive HBase shell.

```
exit
```

10 End the interactive shell in the OpenTSDB container.

```
exit
```

Managing maintenance scripts

The following topics provide information about the maintenance scripts that are installed when Control Center is installed on a host.

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.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.weekly/serviced-zenossdbpack`

This script invokes a `serviced` command weekly, which in turn invokes the database maintenance script for a Zenoss application. If the application is not installed or is offline, the command fails.

This script is required on the master host only.

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 can be removed.

`/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 can be removed from delegate hosts.

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

This script can be removed from delegate hosts.

Control Center configuration variables A

This appendix includes the content in the following list:

Many configuration choices depend on application requirements. Please review your application documentation before configuring hosts.

Removed variables (since version 1.1.1)

The variables in the following table were present in version 1.1.1 and are not present in version 1.2.0. The variables are shown in the order in which they appeared in `/etc/default/serviced`.

Variable	Explanation
TMP	Not POSIX compliant. Replaced by *TMPDIR*.
SERVICED_AGENT	*SERVICED_MASTER* is sufficient to specify the host role.
SERVICED_VARPATH	*SERVICED_ISVCS_PATH*, *SERVICED_VOLUMES_PATH*, and *SERVICED_BACKUPS_PATH* are more specific and preferred.
SERVICED_MONITOR_DFS_MASTER_INTERVAL *SERVICED_MONITOR_DFS_MASTER_RESTART* *SERVICED_MONITOR_DFS_REMOTE_UPDATE_INTERVAL*	Control Center now uses different mechanisms to monitor DFS status.

New variables (since version 1.1.1)

The variables in the following table were not present in version 1.1.1 and are present in version 1.2.0. The variables are shown in the order in which they appear in `/etc/default/serviced`.

Variable	Purpose
TMPDIR	Replacement for *TMP*.
SERVICED_MASTER_IP	The IP address of the master host.
SERVICED_MASTER_POOLID	The name of the resource pool to which the master host belongs.
SERVICED_RPC_TLS_MIN_VERSION *SERVICED_RPC_TLS_CIPHERS*	Additional options for encrypting RPC communications.
SERVICED_UI_POLL_FREQUENCY	The frequency at which browser interface clients poll the server.
SERVICED_MUX_DISABLE_TLS *SERVICED_MUX_TLS_MIN_VER* *SERVICED_MUX_TLS_CIPHERS*	Additional options for encrypting RPC communications.
SERVICED_DM_ARGS *SERVICED_DM_BASESIZE* *SERVICED_DM_LOOPDATASIZE* *SERVICED_DM_LOOPMETADATASIZE* *SERVICED_DM_THINPOOLDEV*	Options for the `devicemapper` storage driver.
SERVICED_STORAGE_STATS_UPDATE_INTERVAL	The number of seconds between polls of kernel statistics about the application data thin pool.
SERVICED_LOGSTASH_CYCLE_TIME	The amount of time between logstash purges.
SERVICED_SVCSTATS_CACHE_TIMEOUT	The lifetime of data in the browser interface cache for statistics about services.
SERVICED_NFS_CLIENT	DEPRECATED: Prevent a delegate host from mounting the DFS. In version 1.2.0, delegate host access to the DFS is controlled through resource pool permissions.
SERVICED_DOCKER_DNS	The Docker Engine DNS configuration for containers.
SERVICED_SNAPSHOT_USE_PERCENT	The amount of free space in the application data thin pool, used to determine whether the pool can store a new snapshot.
SERVICED_ZK_SESSION_TIMEOUT	The number of seconds the ZooKeeper leader waits before flushing an inactive connection.
SERVICED_ES_STARTUP_TIMEOUT	The number of seconds to wait for the Elasticsearch service to start.
SERVICED_RPC_DIAL_TIMEOUT	The number of seconds until an RPC connection attempt times out.
SERVICED_AUTH_TOKEN_EXPIRATION	The lifetime of a delegate host revocation certificate.

Variable	Purpose
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.

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.

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.
- The defaults of the third group may need to be changed before starting the master host for the first time.
- 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* *SERVICED_ALLOW_LOOP_BACK*	The specifications of a `devicemapper`-based application data storage resource for development use.	Provide basic information about the data storage resource.
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. On the master host, the default values of these variables do not need to be changed, unless the Docker registry is hosted on a system other than the master. In that case, *SERVICED_DOCKER_REGISTRY* must be set explicitly on the master host, too.

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

Variable	Endpoint	Description
SERVICED_DOCKER_REGISTRY	*Master-Host*:`5000`	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 cluster 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

Variable	Where to set
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 cluster 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 cluster 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.

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_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. On the master host, the default values of these variables do not need to be changed, unless the Docker registry is hosted on a system other than the master. In that case, *SERVICED_DOCKER_REGISTRY* must be set explicitly on the master host, too.

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

Variable	Endpoint	Description
SERVICED_DOCKER_REGISTRY	*Master-Host*:5000	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 cluster 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 cluster 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 hosts in a Control Center cluster. 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

Variable	Description
	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 Engine 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 cluster 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`.

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. Making a backup copy of the configuration file before editing it 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.

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 Engine 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 cluster 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.

SERVICED_DOCKER_REGISTRY

Default: `{{SERVICED_MASTER_IP}}:5000`

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

The value used to replace `{{SERVICED_MASTER_IP}}` in this variable must match the value of the `--insecure-registry` flag in the `/etc/sysconfig/docker` file.

SERVICED_OUTBOUND_IP

Default: (none)

The default startup routines of `serviced` include attempting to ping `google.com`. When a value is set for this variable, `serviced` does not attempt the ping and assumes it does not have internet access.

Use this variable to specify the IP address of a network interface other than the default, or to prevent `serviced` from assuming it has internet access.

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 cluster hosts.

SERVICED_UI_PORT

Default: `:443`

The port on which the HTTP server listens for requests. The value may be expressed as follows:

- *IP-Address*:*Port-Number*
- :*Port-Number*
- *Port-Number*

All Control Center cluster hosts must have the same value for this variable.

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 cluster 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_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`
15 `TLS_ECDHE_ECDSA_WITH_RC4_128_SHA`
16 `TLS_ECDHE_RSA_WITH_RC4_128_SHA`

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 cluster 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` is supported.

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 Engine 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 ZooKeeper leader waits before flushing an inactive connection.

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 (`,`).

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

Create thin pools on separate block devices:

```
serviced-storage create-thin-pool --size=50G docker /dev/sdb
serviced-storage create-thin-pool --size=200G serviced /dev/sdc
```

Create thin pools on separate partitions:

```
serviced-storage create-thin-pool --size=50% 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
```

Preparing to install without internet access C

This appendix includes procedures for downloading and installing or staging Control Center software and its operating system dependencies. This distribution option is available only to Zenoss customers, to enable creating Control Center deployments without internet access.

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

Procedure	Where to perform
Downloading repository and image files on page 74	A workstation with internet access
Installing the repository mirror on page 75	All Control Center cluster hosts
Staging Docker image files on the master host on page 76	The master host
Staging Docker image files on ZooKeeper ensemble nodes on page 76	Delegate hosts that are ZooKeeper ensemble nodes

Downloading repository and image files

To perform this procedure, you need:

- A workstation with internet access.
- Permission to download files from the *File Portal - Download Zenoss Enterprise Software* site. Zenoss customers may request permission by filing a ticket at the *Zenoss Support* 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 *File Portal - Download Zenoss Enterprise Software* site.
2. Log in with the account provided by Zenoss Support.
3. Download the self-installing Docker image files.

   Select the files with the highest version number. The version number is represented as *Version* in the following file names:

```
install-zenoss-isvcs-zookeeper:v<Version>.run
install-zenoss-serviced-isvcs:v<Version>.run
```

4 Download the Control Center RPM file.

Select the file for the new release:

```
serviced-1.2.3-1.x86_64.rpm
```

5 Download a RHEL/CentOS repository mirror file.

The download site provides a repository mirror file each supported release of RHEL/CentOS. Each file contains the Control Center package and its dependencies.

To download the correct repository mirror file, match the operating system release number in the file name (`centos7.1` or `centos7.2`) with the version of RHEL/CentOS installed on all of the hosts in your Control Center cluster. Also, choose the file with the highest version number. The version number is represented as *Version* in the following file names:

```
yum-mirror-centos7.centos7.1-<Version>.x86_64.rpm
yum-mirror-centos7.centos7.2-<Version>.x86_64.rpm
```

6 Optional: Download the Pacemaker resource agent for Control Center, if necessary.

The resource agent is only needed for high-availability deployments.

```
serviced-resource-agents-<Version>.x86_64.rpm
```

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

- Copy all of the files to the master host.
- Copy the RHEL/CentOS RPM and the Control Center RPM to all delegate hosts.
- Copy the Docker image file for ZooKeeper (`install-zenoss-isvcs-zookeeper:v*.run`) to delegate hosts that are ZooKeeper ensemble nodes.

Installing the repository mirror

Use this procedure to install a local RHEL/CentOS repository mirror on a Control Center host. The mirror contains packages that are required on all Control Center cluster hosts, and its size is approximately 95MB (version 7.2) or 140MB (version 7.1).

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

2 Move the RPM files to `/tmp`.

3 Install the RHEL/CentOS repository mirror.

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

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

4 Copy the Control Center RPM file to the mirror directory.

```
cp /tmp/serviced-*.rpm /opt/zenoss-repo-mirror
```

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

5 Optional: Delete the RPM files, if desired.

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

Staging Docker image files on the master host

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 add Docker image files to the Control Center master 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 Docker image files 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 Docker image files to the Control Center delegate hosts that are ZooKeeper ensemble nodes. The files are used when the ZooKeeper ensemble is configured.

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:v*.run` file to `/root`.
3. Add execute permission to the file.

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

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 hosts in a Control Center cluster.

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.

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, disable time synchronization between guest and host operating systems before performing this procedure.

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 the hosts in an Control Center cluster:

   ```
   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 delegate host in your Control Center cluster.

Note On VMware vSphere guests, disable time synchronization between guest and host operating systems before performing this procedure.

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
```