Control Center Reference Guide

Release 1.3.0

Zenoss, Inc.

www.zenoss.com

Control Center Reference Guide

Part Number: 1340.17.076

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

Contents

About this guide

Control Center Reference Guide provides information and procedures for managing Control Center.

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 (*Control Center Reference Guide*) and part number (1340.17.076) and as much information as possible about the context of your feedback.

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.)
Windows 10	Internet Explorer 11 (Enterprise mode only; compatibility mode is not supported.)
	Firefox 50 and later
	Chrome 54 and later
	Microsoft Edge
Windows Server 2012 R2	Firefox 30
	Chrome 36
Macintosh OS/X 10.9	Firefox 30 and above
	Chrome 36 and above
Ubuntu 14.04 LTS	Firefox 30 and above
	Chrome 37 and above

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

1 Administration reference

This section contains information about and procedures for performing administrative tasks in Control Center.

Control Center application data storage requirements

Control Center uses an LVM thin pool to store tenant (application) data. LVM thin pools include separate storage areas for data and for metadata. For each tenant it manages, Control Center maintains a separate virtual device (a volume) in the data storage area of its LVM thin pool.

To ensure consistency, Control Center requires the following, minimum amount of free space in its thin pool:

- 3GiB available for each tenant volume
- 3GiB available in the data storage area
- 62MiB available in the metadata storage area
- The total amount of metadata storage must be 1% of total data storage

Examining application data storage status

Beginning with release 1.3.0, Control Center initiates an emergency shutdown when the minimum required amounts of free space are not available in the `serviced` thin pool or tenant volumes. For more information, see *Control Center application data storage requirements* on page 6.

Use this procedure to display the amount of free space in a Control Center thin pool and tenant volumes, to determine how much space is available in the LVM volume group that contains the thin pool, and to determine whether additional steps are required.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Display the amount of space available in the `serviced` thin pool.

```
serviced volume status
```

Figure 1: Example output with highlighted values

```
Status for volume /opt/serviced/var/volumes:

Driver:      devicemapper
Driver Type: direct-lvm
Volume Path: /opt/serviced/var/volumes

Thin Pool
---------
Logical Volume:               serviced-serviced--pool
Metadata (total/used/avail):  2.2 GiB / 64.18 MiB (2.85%) / 2.137 GiB (97.15%)
Data (total/used/avail):      200 GiB   / 30.576 GiB (15%)   / 169.424 GiB (85%)

3ir81rg1h8b09ipowynz3qx2f Application Data
------------------------------------------
Volume Mount Point:            /opt/serviced/var/volumes/3ir81rg1h8b09ipowynz3qx2f
Filesystem (total/used/avail): 98.31 GiB / 1.511 GiB (1.5%) / 91.78 GiB (93%)
Virtual device size:           100 GiB
```

The result includes detailed information about the `serviced` thin pool and each tenant volume.

- If the amount of free space in the `serviced` thin pool is sufficient, stop. No further action is required.
- If the amount of free space in the data or metadata portions of the `serviced` thin pool is not sufficient, perform the following steps.

3 Identify the volume group to which the `serviced` thin pool belongs.

```
lvs --options=lv_name,vg_name,lv_size
```

The volume group associated with `serviced-pool` contains the `serviced` thin pool.

4 Display the amount of free space in the volume group that contains the `serviced` thin pool. Replace *Volume-Group* with the name of the volume group identified in the previous step:

```
vgs --no-headings --options=vg_free Volume-Group
```

- If the amount of free space in the volume group is not sufficient to increase one or both of the storage areas of the `serviced` thin pool to their required minimums, add physical or logical storage to the volume group. For more information, refer to your operating system documentation.
- If the amount of free space in the volume group is sufficient to increase one or both of the storage areas of the `serviced` thin pool to their required minimums, proceed to the next step

5 Increase the free space in one or both areas of the `serviced` thin pool.

- To increase the amount of space in the metadata area, proceed to *Adding space to the metadata area of a Control Center thin pool* on page 7.
- To increase the amount of space in the data area, proceed to *Adding space to the data area of a Control Center thin pool* on page 8.
- To increase the size of a tenant volume, which relies on the data area of the thin pool, proceed to *Adding space to a tenant volume* on page 8.

Adding space to the metadata area of a Control Center thin pool

Use this procedure to increase the amount of space in the metadata area of a Control Center thin pool. For more information, see *Control Center application data storage requirements* on page 6.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Display information about LVM logical volumes on the host.

```
lvs --options=lv_name,vg_name,lv_size
```

   Typically, the logical volume is `serviced-pool` and the containing volume group is `serviced`.
3. Add space to the metadata storage area of the `serviced` thin pool.
   In the following command:

   - Replace *Size* with the amount of space to add (in megabytes) and the units identifier (`M`).
   - Replace *Volume-Group* with the name of the LVM volume group identified in the previous step.
   - Replace *Logical-Volume* with the name of the logical volume identified in the previous step.

```
lvextend -L+SizeM Volume-Group/Logical-Volume_tmeta
```

Adding space to the data area of a Control Center thin pool

Use this procedure to increase the amount of space in the data area of a Control Center thin pool. For more information, see *Control Center application data storage requirements* on page 6.

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Display information about LVM logical volumes on the host.

```
lvs --options=lv_name,vg_name,lv_size
```

   Typically, the logical volume is `serviced-pool` and the containing volume group is `serviced`.
3. Add space to the data storage area of the `serviced` thin pool.
   In the following command:

   - Replace *Total-Size* with the sum of the existing device size plus the space to add to the device, in gigabytes. Include the units identifier, `G`.
   - Replace *Volume-Group* with the name of the LVM volume group identified in the previous step.
   - Replace *Logical-Volume* with the name of the logical volume identified in the previous step.

```
lvextend -L+Total-SizeG Volume-Group/Logical-Volume
```

Adding space to a tenant volume

Use this procedure to increase the size of a tenant volume in a Control Center thin pool. For more information, see *Control Center application data storage requirements* on page 6.

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

```
serviced volume status
```

3. Display the device mapper name of the `serviced` thin pool.

```
grep -E '^\b*SERVICED_DM_THINPOOLDEV' /etc/default/serviced \
  | sed -e 's/.*=//'
```

   Typically, the name is `/dev/mapper/serviced-serviced--pool`.
4. Increase the size of the tenant device.

In the following command:

- Replace *Device-Mapper-Name* with the device mapper name of the thin pool.
- Replace *Tenant-ID* with the identifier of the tenant device.
- Replace *Total-Size* with the sum of the existing device size plus the space to add to the device, in gigabytes. Include the units identifier, `G`.

```
serviced-storage resize -d /opt/serviced/var/volumes \
  -o dm.thinpooldev=Device-Mapper-Name Tenant-ID Total-SizeG
```

Using Control Center with a NAT device

Network address translation (NAT) enables one device (a router, switch, or firewall) to connect a local area network with the internet and outside devices. The NAT device forwards traffic to the intended host, and serves as a firewall to systems that are behind the device, making them inaccessible from outside the network.

In a Control Center system without a NAT device, the master host connects directly to the delegate host's `address:rpcport` and requests host information.

When the Control Center master host is outside the network, it can connect to the NAT device, but cannot access delegate hosts behind the device because they have private IP addresses. The NAT device forwards its port to the delegate hosts' `address:rpcport`.

When you add a delegate host by using either Control Center interface (browser or command-line) you must specify the hostname or IP address and port for the NAT device. After you add delegate hosts, you must transfer host keys to the delegate hosts and register them.

The Control Center master host always attempts registration on port 22. If the NAT device forwards port 22 to the delegate host that you are registering, you can remotely register the keys.

```
serviced host add Hostname-Or-IP:Host-Port \
    --nat-address==NAT-Hostname-Or-IP:NAT-Port \
    [-register]
```

If you have two delegates behind the NAT device, change the port forwarding for port 22 between adding the hosts, or transfer the keys file manually to and register from each delegate.

When resetting keys, the CLI supports the `--nat-address` argument. If the delegate is behind a NAT device and port 22 is forwarded to that delegate, you can attempt to register the delegate when resetting the key:

```
serviced key reset Hostname-Or-IP:Host-Port --register \
    --nat-address==NAT-Hostname-Or-IP:NAT-Port
```

Example: Adding delegate hosts to a resource pool

The master host is outside the network. Delegate hosts delegate1 and delegate2 are behind a NAT router. IP address and port information is as follow:

- NAT router: 98.138.252.30
- Delegate1: 192.168.22.100:4979
- Delegate2: 192.168.22.101:4979

The router forwards port 4979 to delegate1's RPC port (4979):

```
serviced host add 192.168.22.100:4979 Resource-Pool \
    --nat-address=98.138.252.30:4979
```

The router forwards port 4980 to delegate2's RPC port (4979):

```
serviced host add 192.168.22.101:4979 Resource-Pool \
    --nat-address=98.138.252.30:4980
```

Security considerations for using Control Centerwith a NAT device

To attach to a service on a delegates behind the NAT device, you must use ssh to access the delegate. From the delegate host, run `serviced service attach`. For security reasons, you cannot use `serviced service attach` from the master to connect to a delegate.

In the Control Center browser interface, for security reasons, you cannot drill down to a service that is running on a delegate behind a NAT device and click **Container Log** for the instance

Backing up and restoring

Use Control Center to back up applications that Control Center manages. Having accurate and tested system backups can mitigate problems caused by software or hardware issues. The Control Center backup process creates a compressed tar archive file (`.tgz`) that can be restored on the same cluster or a similar cluster.

Backups include the current state of the system, the state of all services, configuration information, and application data. The backup process leverages snapshot functionality. Therefore, when a backup is running, you can start and restart services; there is no need to shut down the application or Docker containers. The services are only momentarily suspended to enable reading the data.

You can back up and restore applications by using the browser interface or the command-line interface (CLI). Results are comparable; however, the CLI offers an option to exclude subdirectories from a backup.

The default directory for backup files is `/opt/serviced/var/backups`. The directory can be changed by specifying the *SERVICED_BACKUPS_PATH* environment variable in the Control Center configuration file, `/etc/default/serviced`.

When a full backup is not necessary, such as when you need a checkpoint before installing software, you can perform a snapshot of the system. If you need to revert back to a snapshot, use the rollback feature. You use the CLI to perform snapshot and rollback.

With both backup and snapshot, Control Center

- Creates a tag for the Docker image of each service with metadata about the application data.
- Creates a separate snapshot of the LVM thin pool, which stores both application data and snapshots of the application data.

When you create a backup, Control Center also exports the snapshots to an archive file and moves them out of the LVM thin pool. Backups do not affect time-to-live (TTL).

For more information, see *Snapshot and rollback* on page 13 and *Command-line interface reference* on page 24.

Best practices for backup and restore

Review considerations and best practices that apply to application backup and restore.

- Ensure that you have enough free space to receive and store backups. Running low on available disk space results in errors and affects system performance.
- To provide a historical archive, back up on a regular schedule. Back up as needed when you perform less-frequent tasks such as moving data from one instance to another or duplicating an instance for testing or failover purposes.
- Before upgrading or testing an application, ensure that you have a recent backup that successfully restores.
- Regularly back up the production environment and potentially the system from the initial deployment.
- Back up to a nonactive resource target, such as a separate disaster recovery system or test environment.
- Store backups on a machine other than the Control Center master.
- Copy or migrate backups to an off-system location for safekeeping and to help regulate storage space usage on the master.
- Backups that were created using Control Center 1.0.x cannot be restored in Control Center 1.1.x or later.
- In Control Center 1.2.x and later, you can restore backups that were created using Control Center 1.1.x or later.
- You can restore a backup to the system on which it was created or to an alternate system. When restoring a backup from one system to an alternate system, ensure that
  - The alternate system mirrors at least one device from the backed-up system.
  - Services that were added to the alternate system by a previous restore have been manually deleted.
- Frequently test restoring from a backup to ensure that the backup restores successfully, and that the restored system is an accurate representation of the state of the deployment when the backup was performed.
- Restoring from a backup file does not remove services that were added after taking the backup. That is, if you create a backup, add a service, and then restore from the backup, the service is not deleted as part of the restore process.
- If an outage occurs during a restore from a backup, you can resume the restore because Control Center preserves complete data on the system. For example, if two of six backed up snapshots are restored before an outage, when you resume the restore, those two snapshots are saved on the system, and are not downloaded again.

Backing up using the browser interface

Using the browser interface, you can create a backup of your entire system.

1. Log in to the Control Center browser interface.
2. Click the **Backup / Restore** tab.
3. Click **Create Backup**.
4. At the prompt, confirm your selection by clicking **Create Backup**.

When the backup is complete, the compressed archive file name is displayed.

Backing up using the CLI

As an alternative to performing application backup from the Control Center browser interface, you can use the command-line interface (CLI).

Using the CLI, you can back up your entire system or specify one or more tenant volumes to exclude from the backup. For example, to save resources, you might exclude the tenant volume that contains performance data, `hbase-master`. If you want to create a backup to restore on another system, you might exclude the tenant volumes for the events database and index, `mariadb-events` and `zeneventserver`.

The default directory for backup files is `/opt/serviced/var/backups`. The directory can be changed by specifying the *SERVICED_BACKUPS_PATH* environment variable in the Control Center configuration file, `/etc/default/serviced`.

Backing up the entire system

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Start the backup:

```
serviced backup /opt/serviced/var/backups
```

After a successful backup, the system displays the name of the backup file.

Example result:

```
backup-2017-03-07-203717.tgz
```

Exclude one tenant volume from the backup

By default, Control Center stores application data in `/opt/serviced/var/volumes`. The directory can be changed by specifying the `SERVICED_VOLUMES_PATH` environment variable in the Control Center configuration file, `/etc/default/serviced`. If necessary, replace `/opt/serviced/var/volumes` with your path.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Display the Control Center tenant identifier.

```
ls /opt/serviced/var/volumes
```

Example result:

```
cvs0ul2tmvjcitolrm7p0d8bx
```

3. Display the directories under the Control Center tenant identifier.
   Replace *Tenant-ID* with the identifier displayed in the previous step.

```
ls /opt/serviced/var/volumes/Tenant-ID
```

4. Exclude the tenant volume from the backup.
   For example, exclude `hbase-master`.

```
serviced backup /opt/serviced/var/backups --exclude hbase-master
```

If you use automated backups, edit the scripts to exclude tenant volumes.

Exclude multiple tenant volumes from the backup

By default, Control Center stores application data in `/opt/serviced/var/volumes`. The directory can be changed by specifying the `SERVICED_VOLUMES_PATH` environment variable in the Control Center configuration file, `/etc/default/serviced`. If necessary, replace `/opt/serviced/var/volumes` with your path.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Display the Control Center tenant identifier.

```
ls /opt/serviced/var/volumes
```

Example result:

```
cvs0ul2tmvjcitolrm7p0d8bx
```

3 Display the directories under the Control Center tenant identifier.

Replace *Tenant-ID* with the identifier displayed in the previous step.

```
ls /opt/serviced/var/volumes/Tenant-ID
```

4 Exclude multiple tenant volumes from the backup.

For example, exclude `mariadb-events` and `zeneventserver`.

```
serviced backup /opt/serviced/var/backups --exclude mariadb-events \
   --exclude zeneventserver
```

Restoring from a backup

Restore an instance of an application from a backup file on the same system, or restore from a backup file to duplicate an instance on a new, similarly configured cluster.

If you are restoring from a backup that was created on another system, copy the backup archive file to the target system.

1 Log in to the Control Center browser interface.
2 In the **Applications** table, identify the name of the application instance.
3 Stop the instance and verify that its subservices are stopped.
   a In the **Actions** column of the **Applications** table, click **Stop**.
   b In the **Stop Service** dialog box, click **Stop Service and Children**.
   c In the **Applications** column of the **Applications** table, click the name of the stopped instance, and then scroll down to the **Services** table to verify that all services are stopped.

   Because snapshots are loaded to disk, during a restore you are not *required* to stop services while the file is loaded. Though the restore will not succeed, the snapshot and images are available for you to manually rollback each application.
4 Click the **Backup / Restore** tab.
5 Beside the backup file that you want to use to restore your application, click **Restore Backup**. Confirm your selection by clicking **Restore**.
6 When the restore is finished, click the **Applications** tab, then click **Start** beside the instance you just restored.
7 Review and if necessary, define IP assignments.
   a Click **Applications** and then click the application instance.
   b Review the **IP Assignments** table. If all services have an IP assignment, no action is required.
   c For any service that does not have an automatic IP assignment, click **Assign**, choose an IP, and then click **Assign IP**.

Snapshot and rollback

Though backups are the most reliable and durable way to preserve Docker images and configurations, creating a backup of an entire application is not always practical. However, you need to safeguard against potential risk when changing the system. In these cases, you can create a snapshot of the system.

Snapshot functionality provides a time- and space-efficient method of copying data. Create a snapshot whenever you need a save point for Docker images, such as before committing container changes.

With both snapshot and backup, Control Center

- Creates a tag for the Docker image of each service with metadata about the application data.
- Creates a separate snapshot of the LVM thin pool, which stores both application data and snapshots of the application data.

Snapshots are intended to serve as short-term save points only, and therefore have a default time-to-live (TTL) value of 12 hours. If you need to keep a snapshot beyond the TTL, tag the snapshot to prevent it from being deleted after the TTL expires. For historical backups of data that you need to save long-term, create full backups instead of snapshots.

You can use the rollback functionality to go back to a snapshot image. For example, roll back if changes to an application cause a failure or other degradation. Rolling back returns the application and distributed file system to the state that existed at the time of the snapshot.

Note Rolling back from a snapshot does not remove services that you added after creating the snapshot. That is, if you create a snapshot, add a service, and then roll back, the service remains on the system; it is not deleted as part of the roll back.

Control Center uses *thin provisioning*, which enables it to create snapshots of the application data volume. Thin provisioning is a virtualization method that allocates data blocks only when data is written (copy-on-write).

Because snapshots track changes to the file system over time, their space requirements expand incrementally as application data changes. Application data and snapshots share the same base device; therefore, ensure that snapshots do not fill up the base device storage. For information about extending storage, see *Control Center application data storage requirements* on page 6.

Creating a snapshot

1. Log in to the Control Center host as a user with `serviced` CLI privileges.
2. Find the identifier of the service; for example, `Zenoss.resmgr`.

   ```
   serviced service list
   ```
3. Create the snapshot.

   Replace *SERVICEID* with the identifier of the service.

   ```
   serviced snapshot add SERVICEID
   ```
4. Verify the existence of the snapshot.

   ```
   serviced snapshot list
   ```
5. To keep the snapshot for longer than the default 12-hour TTL, tag it.

   Replace *SNAPSHOTID* with the identifier of your snapshot and *TAG-NAME* with your text.

   ```
   serviced snapshot tag SNAPSHOTID TAG-NAME
   ```
6. To make a snapshot subject to the TTL value, untag it.

   Replace *SNAPSHOTID* with the identifier of your snapshot and *TAG-NAME* with your text.

   ```
   serviced snapshot untag SNAPSHOTID TAG-NAME
   ```

Rolling back to a snapshot

Before rolling back, you must stop services that are used in the snapshot image. The following procedure includes this step.

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

2 To roll back to a snapshot, you must find the identifier of the snapshot.

```
serviced snapshot list
```

3 Roll back to the snapshot.

Replace *SNAPSHOTID* with the identifier of your snapshot. The `--force-restart` flag automatically stops the affected services before rollback and starts them after completion.

```
serviced snapshot rollback SNAPSHOTID --force-restart
```

Stopping and starting Control Center for maintenance

Before performing maintenance, such as operating system upgrades or applying patches, properly stop and start Control Center. This section provides procedures for single-host and multi-host deployments.

Stopping Control Center (single-host deployment)

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

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

2 Stop the top-level service `serviced` is managing, if necessary.

a Show the status of running services.

```
serviced service status
```

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

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

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

```
serviced service stop Service
```

c Monitor the stop.

```
serviced service status
```

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

3 Stop the Control Center service.

```
systemctl stop serviced
```

4 Ensure that no containers remain in the local repository.

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

```
docker ps -qa
```

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

b Remove all remaining containers.

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

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

```
docker ps -qa
```

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

d Disable the automatic startup of `serviced`.

```
systemctl disable serviced
```

e Reboot the host.

```
reboot
```

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

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Starting Control Center (single-host deployment)

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

1. Log in to the master host as `root`, or as a user with superuser privileges.
2. Determine whether `serviced` is configured to start when the system starts.

```
systemctl is-enabled serviced
```

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

```
systemctl enable serviced
```

3. Start the Control Center service.

```
systemctl start serviced
```

4. Optional: Monitor the startup, if desired.

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

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

Stopping Control Center (multi-host deployment)

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

Stopping a master host (multi-host deployment)

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

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

      ```
      serviced service status
      ```

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

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

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

      ```
      serviced service stop Service
      ```

   c. Monitor the stop.

      ```
      serviced service status
      ```

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

   ```
   systemctl stop serviced
   ```

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

      ```
      docker ps -qa
      ```

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

   b. Remove all remaining containers.

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

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

      ```
      docker ps -qa
      ```

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

   d. Disable the automatic startup of `serviced`.

      ```
      systemctl disable serviced
      ```

e Reboot the host.

```
reboot
```

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

g Enable the automatic startup of `serviced`.

```
systemctl enable serviced
```

Stopping a delegate host

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

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

2 Stop the Control Center service.

```
systemctl stop serviced
```

3 Ensure that no containers remain in the local repository.

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

```
docker ps -qa
```

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

b Remove all remaining containers.

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

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

c Stop the NFS and Docker 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
```

Starting Control Center (multi-host deployment)

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

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

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

```
systemctl is-enabled serviced
```

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

```
systemctl enable serviced
```

3 Identify the hosts in the ZooKeeper ensemble.

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

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

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

5 On all ensemble hosts, start `serviced`.

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

```
systemctl start serviced
```

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

a Attach to the container of the ZooKeeper service.

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

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

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

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

c Query the other delegate hosts to identify their role in the ensemble.

Replace *Delegate* with the hostname or IP address of a delegate host:

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

d Detach from the container of the ZooKeeper service.

```
exit
```

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

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

```
systemctl start serviced
```

8 Optional: Monitor the startup, if desired.

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

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

Emergency shutdown of services

Control Center monitors each service's short-term storage usage trends and current usage levels. When Control Center predicts that a service is about to exhaust storage space, it initiates an automatic emergency shutdown of the service. By shutting down while enough space is available to perform recovery operations, Control Center minimizes the risk of data corruption.

Emergency shutdown is performed for services that are in a resource pool that has DFS access. Services that are in pools that do not have DFS permissions and do not write to the DFS continue running.

Control Center displays thin pool and DFS information in the following graphs in the browser interface:

- On the **Applications** tab, the **Thin Pool Usage** graph shows used and available bytes for the thin pool.
- On the **Applications** page for each service, the **DFS Usage** graph shows used and available bytes for the DFS.

By comparing usage to the available storage and the amount of space that must be reserved, Control Center determines when a service must be shut down before filling the thin pool or DFS, and initiates the emergency shutdown. The browser interface identifies services in the emergency shutdown state, as does issuing the `serviced service status` command in the command line.

To minimize data loss, Control Center shuts down services in the following order: databases; services that cannot be recovered; indices and services that are difficult to recover; any other services. Services in emergency shutdown status cannot be restarted until the underlying cause of the shutdown is resolved.

To resolve an emergency shutdown:

1 Examine the service that was shut down to determine why it was using excessive storage and correct the issue. For example:

- If an application was writing a large amount of performance data to the tenant device, add space to the device. See *Control Center application data storage requirements* on page 6.
- If too many snapshots are stored on the device, delete those that you no longer need. See *Snapshot and rollback* on page 13.

- If a usage anomaly might have occurred, wait for usage levels to return to normal.

2. Clear the emergency shutdown flags.
3. Start the service by using the browser interface or command line interface. Control Center starts services in the reverse order of shutdown.

Resetting emergency shutdown flags

After resolving the issue that caused an emergency shutdown, use this procedure to service status, and then restart the service.

1. Log in to the Control Center master host as `root`, or as a user with superuser privileges.
2. Check services for the emergency shutdown flag:

```
for s in $(serviced service list --show-fields ServiceID); \
               do serviced service list $s | grep EmergencyShutdown\";
 done;
```

"True" indicates an emergency shutdown.

3. Clear the emergency-shutdown flag for a service or the entire application:

```
serviced service clear-emergency SERVICEID
```

```
serviced service clear-emergency APPLICATION_ID
```

4. Check services for the emergency shutdown flag:

```
for s in $(serviced service list --show-fields ServiceID); \
               do serviced service list $s | grep EmergencyShutdown\";
 done;
```

"False" indicates that the flags have been cleared.

5. Start the service:

```
serviced service start SERVICEID
```

Rolling restart of services

To reduce or eliminate downtime for services with multiple instances, Control Center restarts instances of the service one at a time.

The Control Center browser interface visually indicates when a service is restarting and whether an instance is down during the restart. When a wide area network (WAN) outage occurs, the rolling restart proceeds, with instances on the disconnected hosts restarting when the WAN is restored.

The serial process applies to *restart* only. When stopping and starting services, Control Center starts and stops all instances immediately.

Optionally, you can specify that Control Center is to stop all instances of a service before restarting.

Changing rolling restart

Use this procedure to stop all instances of a service before restarting, instead of performing a rolling restart.

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

2 Restart the service with the `--rebalance` option:
Replace *Service* with the name or identifier of the service:

```
serviced service restart --rebalance Service
```

Command-line interface reference 2

This section contains information about and procedures for using the Control Center command-line interface (CLI). The CLI uses the same application programming interface that the browser interface uses. To perform many actions, you can use either interface.

Invoking serviced

To use `serviced`, you need a login account on hosts in Control Center pools. The account must be a member of the `docker` group.

You can invoke `serviced` from the Control Center master host or a delegate host. For example, to list the services running on the master host, enter the following command:

```
serviced service list
```

To invoke `serviced` on a delegate host, you can specify the master host and port by using the `--endpoint` global option. Replace *Master-Host* with the hostname or IP address of the Control Center master host.

Example:

```
serviced --endpoint Master-Host:4979 service list
```

Note To commit a container, you must run a `serviced` CLI session on the Control Center master host.

serviced

NAME

`serviced` - A container-based management system.

SYNTAX

`serviced [global options] command [command options] [arguments...]`

DESCRIPTION

`serviced` is an open-source, application service orchestrator based on *Docker*.

GLOBAL OPTIONS

--docker-registry *Master-Hostname*:5000

The local Docker registry to use. For more information, see *SERVICED_DOCKER_REGISTRY* in the section about configuration file environment variables.

--static-ip *IP-Address* `[`**--static-ip *IP-Address***`]` **…**

One or more static IP addresses for a `serviced` instance to advertise. For more information, see *SERVICED_STATIC_IPS* in the section about configuration file environment variables.

--endpoint *Host*:*Port*

The `serviced` RPC endpoint. The value of *Host* is the hostname or IP address of the master host. The default value of *Port* is 4979. For more information, see *SERVICED_ENDPOINT* in the section about configuration file environment variables.

--outbound *IP-Address*

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 option to specify the IP address of a network interface other than the default, or to prevent `serviced` from assuming it has internet access. For more information, see *SERVICED_OUTBOUND_IP* in the section about configuration file environment variables.

--uiport:*Port*

The port on which the HTTP server listens for requests. The default value is 443, unless *SERVICED_UI_PORT* is set in the configuration file. For more information, see the section about configuration file environment variables.

--nfs-client *value*

Determines whether a `serviced` delegate mounts the DFS. The default value is 1 (enable) unless *SERVICED_NFS_CLIENT* is set in the configuration file. For more information, see the section about configuration file environment variables.

Note Before changing the default, ensure that no stateful services can run on the host. Disabling the DFS can destroy application data. To disable mounting, set the value to `0`.

--listen:*Port*

The `serviced` RPC endpoint on the local host. The default value of *Port* is 4979.

--docker-dns *Option* `[`**--docker-dns *Option***`]` **…**

One or more DNS configuration flags for Docker to use when starting containers.

--master

Run the application services scheduler and other internal services.

--agent

Run application services scheduled by the master.

--mux *Port*

The port used for traffic among Docker containers. The default value is 22250, unless *SERVICED_MUX_PORT* is set in the configuration file. For more information, see the section about configuration file environment variables.

--mux-disable-tls

Determines whether inter-host traffic among Docker containers is encrypted with TLS. Intra-host traffic among Docker containers is not encrypted.

The default value is 0 (enabled) unless *SERVICED_MUX_DISABLE_TLS* is set to `1` (disable encryption) in the configuration file.

--mux-tls-ciphers *Option* `[`**--mux-tls-ciphers *Option***`]…`

The list TLS ciphers `serviced` supports for mux traffic. The default may be set in by *SERVICED_MUX_TLS_CIPHERS* in the configuration file. For more information, see the section about configuration file environment variables.

--mux-tls-min-version

The minimum version of TLS that `serviced` accepts for mux traffic. Valid values are `VersionTLS11` and `VersionTLS12`. The default value is `VersionTLS11` unless *SERVICED_MUX_TLS_MIN_VERSION* is set in the configuration file. For more information, see the section about configuration file environment variables.

--var *Path*

This option has been deprecated.

volumes-path *Path*

The location of `serviced` application data. The default path is `/opt/serviced/var/volumes` unless the value of the *SERVICED_VOLUMES_PATH* is set in the configuration file. For more information, see the section about configuration file environment variables.

isvcs-path *Path*

The location of `serviced` internal services data. The default path is `/opt/serviced/var/isvcs` unless the value of the *SERVICED_ISVCS_PATH* is set in the configuration file. For more information, see the section about configuration file environment variables.

backups-path *Path*

The location of `serviced` backup files. The default path is `/opt/serviced/var/backups` unless the value of the *SERVICED_BACKUPS_PATH* is set in the configuration file. For more information, see the section about configuration file environment variables.

etc-path *Path*

The location of `serviced` configuration files. The default path is `/opt/serviced/etc`.

--keyfile *Path*

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`.

The default value is *`$TMPDIR`*`/zenoss_key.[0-9]+`, unless *SERVICED_KEY_FILE* is set in the configuration file. For more information, see the section about configuration file environment variables.

--certfile *Path*

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`.

The default value is *`$TMPDIR`*`/zenoss_cert.[0-9]+`, unless *SERVICED_CERT_FILE* is set in the configuration file. For more information, see the section about configuration file environment variables.

--zk *Host*:*Port* `[`**--zk *Host*:*Port***`]…`

One or more ZooKeeper endpoints. If multiple endpoints are specified, `serviced` tries each in turn until it connects to a working server. The default may be set by *SERVICED_ZK* in the configuration file. For more information, see the section about configuration file environment variables.

--mount *Option* `[`**--mount *Option***`]…`

One or more bind mounts for a container. The syntax for *Option* is `DOCKER_IMAGE,HOST_PATH[,CONTAINER_PATH]`.

--fstype *Driver*

The driver to manage application data storage on the `serviced` master host. The default is `devicemapper` unless *SERVICED_FS_TYPE* is set in the configuration file. For more information, see the section about configuration file environment variables.

--alias *Alias* `[`**--alias *Alias***`]…`

One or more DNS aliases to associate with a container.

--es-startup-timeout *Duration*

The number of seconds to wait for Elasticsearch to complete its startup. The default value is 600 seconds (10 minutes).

--max-container-age *Duration*

The number of seconds `serviced` waits before removing a stopped container. The default value is 86400 seconds (24 hours), unless *SERVICED_MAX_CONTAINER_AGE* is set in the configuration file. For more information, see the section about configuration file environment variables.

--max-dfs-timeout *Duration*

The number of seconds to wait for a snapshot to complete. The default value is 300 seconds (5 minutes).

--virtual-address-subnet *Subnet*

The private subnet for containers that use virtual IP addresses on a host. The default value is 10.3.0.0/16, unless *SERVICED_VIRTUAL_ADDRESS_SUBNET* is set in the configuration file. For more information, see the section about configuration file environment variables.

--master-pool-id *Pool-ID*

The name of the resource pool to which the `serviced` instance configured as master belongs. The default value of *Pool-ID* is `default`.

--admin-group *Group*

The name of the Linux group on the `serviced` master host whose members are authorized to use the `serviced` browser interface. The default is `wheel` unless *SERVICED_ADMIN_GROUP* is set in the configuration file. For more information, see the section about configuration file environment variables.

--storage-opts *Option* `[`**--storage-opts *Option***`]…`

Storage arguments to initialize the filesystem.

--isvcs-start *Option* `[--isvcs-start option` ***Option***`]…`

Enables one or more internal services to run on a delegate host. Currently, only `zookeeper` is supported. If *SERVICED_ISVCS_START* is set in the configuration file, its value is used. For more information, see the section about configuration file environment variables.

--isvcs-zk-id *Indentifier*

The unique identifier (a positive integer) of a ZooKeeper ensemble node. If *SERVICED_ISVCS_ZOOKEEPER_ID* is set in the configuration file, its value is used. For more information, see the section about configuration file environment variables.

--isvcs-zk-quorum *Option* [--isvcs-zk-quorum *Option*`]…`

The list of nodes in a ZooKeeper ensemble. If *SERVICED_ISVCS_ZOOKEEPER_QUORUM* is set in the configuration file, its value is used. For more information, see the section about configuration file environment variables.

--tls-ciphers *Option* `[`**--tls-ciphers *Option***`]…`

The list TLS ciphers that `serviced` accepts for HTTP traffic. If *SERVICED_TLS_CIPHERS* is set in the configuration file, its value is used. For more information, see the section about configuration file environment variables.

--tls-min-version *Version*

The minimum version of TLS that `serviced` accepts for HTTP traffic. Valid values include the default, `VersionTLS11`, and `VersionTLS12`. If *SERVICED_TLS_MIN_VERSION* is set in the configuration file, its value is used. For more information, see the section about configuration file environment variables.

--report-stats

Enable reporting statistics in a container.

--host-stats *Host*:*Port*

The endpoint of the `serviced` metrics consumer service. The default value of *Host* is the IP address of the master host, and the default value of *Port* is 8443. If *SERVICED_STATS_PORT* is set in the configuration file, its value is used instead of the default endpoint. For more information, see the section about configuration file environment variables.

--stats-period *Duration*

The frequency, in seconds, at which delegates gather metrics to send to the `serviced` metrics consumer service on the master host. The default value of *Duration* is 10, unless *SERVICED_STATS_PERIOD* is set in the configuration file. For more information, see the section about configuration file environment variables.

--mc-username *User*

The username of the OpenTSDB account that MetricConsumer uses gain access to data stored by `serviced`.

--mc-password *Password*

The password of the OpenTSDB account that MetricConsumer uses gain access to data stored by `serviced`.

--cpuprofile

Instructs a container to write its CPU profile to a file.

--isvcs-env *Option* `[`**--isvcs-env *Option***`]…`

Startup arguments to pass to internal services. The default value is no arguments, unless *SERVICED_ISVCS_ENV_[0-9]+* is set in the configuration file. For more information, see the section about configuration file environment variables.

--debug-port *Port*

The port on which `serviced` listens for HTTP requests for the *Go profiler*. The default value of *Port* is 6006, unless *SERVICED_DEBUG_PORT* is set in the configuration file. To stop listening for requests, set the value to `-1`. For more information, see the section about configuration file environment variables.

--max-rpc-clients *Count*

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.

The default value is 3, unless *SERVICED_MAX_RPC_CLIENTS* is set in the configuration file. For more information, see *SERVICED_MAX_RPC_CLIENTS* in the section about configuration file environment variables.

--rpc-dial-timeout *Duration*

The number of seconds `serviced` waits before giving up on attempts to connect to the RPC endpoint on the master host.

--rpc-cert-verify *Value*

Determines whether `serviced` is enabled to perform TLS certificate verification for RPC connections. The default value is `false` (disabled) unless *SERVICED_RPC_CERT_VERIFY* is set in the configuration file. For more information, see the section about configuration file environment variables.

--rpc-disable-tls *Value*

Determines whether `serviced` enabled to encrypt RPC traffic with TLS. The default value is `false` (disabled) unless *SERVICED_RPC_DISABLE_TLS* is set in the configuration file. For more information, see the section about configuration file environment variables.

--rpc-tls-ciphers *Option* `[`**--rpc-tls-ciphers*Option***`]` **…**

The list of TLS ciphers `serviced` prefers for RPC connections. If *SERVICED_RPC_TLS_CIPHERS* is set in the configuration file, its value is used. For more information, see the section about configuration file environment variables.

--rpc-tls-min-version *Version*

The minimum version of TLS `serviced` accepts for RPC connections. Valid values include the default, `VersionTLS11`, and `VersionTLS12`. The default value is `VersionTLS10` unless *SERVICED_RPC_TLS_MIN_VERSION* is set in the configuration file. For more information, see the section about configuration file environment variables.

--snapshot-ttl *Duration*

The number of hours an application data snapshot is retained before removal. The default value is 12 unless *SERVICED_SNAPSHOT_TTL* is set in the configuration file. For more information, see the section about configuration file environment variables.

--snapshot-space-percent *Value*

The amount of free space in the thin pool, expressed as a percentage the total size. This value is used to determine whether the thin pool can hold a new snapshot. The default value is 20 unless *SERVICED_SNAPSHOT_USE_PERCENT* is set in the configuration file. For more information, see the section about configuration file environment variables.

--controller-binary *Path*

The path to the container controller binary. The default is `/opt/serviced/bin/serviced-controller`.

--log-driver *file*

The log driver for all Docker container logs, including containers for Control Center internal services.

--log-config *Option* `[`**--log-config*Option***`]` **…**

A list of Docker `--log-opt` options as *`key=value`* pairs.

--ui-poll-frequency *Duration*

The number of seconds between polls from browser interface clients. The value is included in a JavaScript library that is sent to the clients. The default value is 3 unless *SERVICED_UI_POLL_FREQUENCY* is set in the configuration file. For more information, see the section about configuration file environment variables.

--storage-stats-update-interval *Duration*

The frequency in seconds that the thin pool usage is analyzed. The default value is 300 (five minutes) unless *SERVICED_STORAGE_STATS_UPDATE_INTERVAL* is set in the configuration file. For more information, see the section about configuration file environment variables.

--zk-session-timeout *Duration*

The number of seconds the ZooKeeper leader waits before flushing an inactive connection. The default value is 15 unless *SERVICED_ZK_SESSION_TIMEOUT* is set in the configuration file. For more information, see the section about configuration file environment variables.

--auth-token-expiry

The expiration time, in seconds, of delegate authentication tokens. The default value is 3600 (one hour) unless *SERVICED_AUTH_TOKEN_EXPIRATION* is set in the configuration file. For more information, see the section about configuration file environment variables.

--logtostderr

Write log messages to STDERR instead of the system log.

--alsologtostderr

Write log messages to STDERR as well as the system log.

--logstashurl *Host*:*Port*

The endpoint of the logstash service. The default value of *Host* is the IP address or hostname of the `serviced` master host and the default value of *Port* is 5042. If *SERVICED_LOG_ADDRESS* is set in the configuration file, its value is used instead of the default endpoint. For more information, see the section about configuration file environment variables.

--logstash-es *Host*:*Port*

The endpoint of the logstash Elasticsearch service. The default value of *Host* is the IP address of the master host, and the default value of *Port* is 9100. If *SERVICED_LOGSTASH_ES* is set in the configuration file, its value is used instead of the default endpoint.

--logstash-max-days *Duration*

The maximum number of days to keep application logs in the logstash database before purging them. The default value of *Duration* is 14, unless *SERVICED_LOGSTASH_MAX_DAYS* is set in the configuration file. When this argument and `--logstash-max-size` are used at the same time, both conditions are evaluated and enforced. For more information, see the section about configuration file environment variables.

--logstash-max-size *Quantity*

The maximum size of the logstash database, in gigabytes. When this argument and `--logstash-max-days` are used at the same time, both conditions are evaluated and enforced. The default value of *Quantity* is 10, unless *SERVICED_LOGSTASH_MAX_SIZE* is set in the configuration file. For more information, see the section about configuration file environment variables.

--logstash-cycle-time *Duration*

The amount of time between logstash purges, in hours. The default value is 6 unless *SERVICED_LOGSTASH_CYCLE_TIME* is set in the configuration file. For more information, see the section about configuration file environment variables.

--v *Level*

The log level `serviced` uses when writing to the system log. Valid values are `0` (normal) and `2` (debug). The default value is 0, unless *SERVICED_LOG_LEVEL* is set in the configuration file. For more information, see the section about configuration file environment variables.

--stderrthreshold *Level*

Write log messages at or above *Level* to STDERR, in addition to the system log. The value of *Level* may be 0 (INFO), 1 (WARNING), 2 (ERROR), or 3 (FATAL). The default value is 2.

--vmodule

Module-specific logging. For more information, refer to the *Google Logging* documentation.

--log_backtrace_at *File*:*Line*

Emit a stack trace when logging hits the specified line and file.

--config-file ***Path***

The path of the configuration file. The default is `/etc/default/serviced`.

--allow-loop-back ***Value***

Determines whether loop-back files can be used with the `devicemapper` storage driver. This option is not supported for production use.

--version

Display minimal version information about the `serviced` binary. To display additional information, use the `serviced version` command.

[**--help** | **--h**]

Display help information.

COMMANDS

`backup`

Copy all templates, services, and application data into a compressed `tar` archive file.

`config`

Report on the `serviced` configuration.

`debug`

Manage debugging.

`docker`

Docker administration commands.

`healthcheck`

Report on the health of `serviced`.

`[help|h]`

Display a global or command-specific help message.

`host`

Administer hosts.

`key`

Display the host's public key.

`log`

Administer logs.

`metric`

Administer metrics.

`pool`

Administer resource pool data.

`restore`

Reconstruct templates, services, and application data from a compressed `tar` archive file created with `backup`.

`script`

Verify or perform the commands in a script file.

`service`

Administer services.

`snapshot`

Administer snapshots.

`template`
Administer templates.

`version`
Display `serviced` version information.

`volume`
Administer volume data.

INVOCATION

Service (daemon) control commands include `start`, `stop` and `reload`. The `reload` command sends `SIGHUP` to the daemon, which restarts all internal services except ZooKeeper.

```
sudo systemctl [start|stop|reload] serviced
```

MISCELLANEOUS

Sending `SIGUSR1` to the `serviced` process toggles the log level between 0 and 2.

To attach to a container running on a remote host, log in to the container from the `serviced` master host. If you are running a Linux shell on a delegate host, you can specify the `--endpoint` option in the `serviced` invocation.

`serviced` relies on Docker, and some administration procedures include `docker` commands. However, commands that manipulate containers directly, such as `docker pause`, should not be used when `serviced` is running.

During installation, `serviced` creates the internal services directory on the master host, so `serviced` commands must be run as `root`, or as a user with superuser privileges. After the master host is added as a delegate, `serviced` commands use the delegate host authorization keys, so root is no longer required.

ENVIRONMENT

SERVICED_HOME
The install path of `serviced`. The default value is `/opt/serviced`.

FILES

`/etc/default/serviced`

serviced backup

The `serviced backup` command saves a snapshot of the current state of the system, the state of all services, and application data to a compressed tar archive file (`.tgz`).

You can backup the entire system, or exclude certain directories, such as a directory that contains application performance data.

USAGE

```
serviced backup [command options] [arguments...]
```

OPTIONS

--exclude '--exclude option --exclude option'

Subdirectory of the tenant volumes to exclude from a backup.

--help, -h

Show the help for an option.

serviced docker

The `serviced docker` command administers Docker images and registry.

USAGE

```
serviced docker [global options] command [command options] [arguments...]
```

COMMANDS

The following commands are available for `serviced docker`:

sync

Asynchronously push all images from the serviced Docker registry index into the Docker registry.

reset-registry

For upgrades only, download the latest images from the Docker registry and save them into the serviced Docker registry index.

migrate-registry

Migrate Docker registry data into another remote registry.

override

Replace an image in the Docker registry with a new image

help, h

Show a list of commands or the help for a single command.

Usage

```
serviced docker
```

OPTIONS

--generate-bash-completion

--help, -h

Shows the help for an option.

serviced host

The `serviced host` command administers host data.

USAGE

```
serviced host [global options] command [command options] [arguments...]
```

COMMANDS

The following commands are available for `serviced host`:

list

Lists all hosts.

add

Adds a host.

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

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

remove, rm

Removes a host.

register

Sets the authentication keys to use for a host. When `KEYSFILE` is `-`, keys are read from standard input (stdin).

set-memory

Sets the memory allocation for a specific host.

help, h

Shows a list of commands or the help for a single command.

OPTIONS

--generate-bash-completion

--help, -h

Shows the help for an option.

serviced key

The `serviced key` command displays the host's public key.

USAGE

```
serviced key [global options] command [command options] [arguments...]
```

COMMANDS

The following commands are available for `serviced key`:

list

Shows the public key for the host.

reset

Regenerates the public key for the host.

help, h

Show a list of commands or the help for a single command.

OPTIONS

--generate-bash-completion

--help, -h
Shows the help for an option.

serviced log export

The `serviced log export` command exports application log files from Elasticsearch for one or more services based on service identifier or service name.

USAGE

```
command export [command options] [arguments...]
```

OPTIONS

--from
Specify the start date.

--to
Specify the end date.

--service '--service option --service option'
Specify the service ID or service name. Includes all subservices.

--out
Specify the path to the output files.

--debug, -d
Show additional diagnostic messages.

EXAMPLE

```
serviced log export --service SERVICE_ID1 --service SERVICE_ID2
```

serviced pool

Use the `serviced pool` command to view and manage Control Center resource pools.

USAGE

```
serviced pool [global options] command [command options] [arguments...]
```

OPTIONS

--generate-bash-completion

--help, -h
Show the help for an option.

COMMANDS

The following commands are available for `serviced pool`:

list
List all pools.

add
Add a new resource pool.

remove, rm
Remove an existing resource pool.

list-ips
Lists the IP addresses for a resource pool.

add-virtual-ip
Add a virtual IP address to a resource pool.

remove-virtual-ip
Remove a virtual IP address from a resource pool.

set-conn-timeout
Set a connection timeout for a high-latency resource pool (for example, 5m, 2h, 6.6s).

set-permission
Set permission flags for hosts in a resource pool.

help, h
Show a list of commands or the help for a single command.

serviced pool set-conn-timeout

The `serviced pool set-conn-timeout` command sets the length of time the scheduler waits for a disconnected delegate to rejoin its pool before moving the services scheduled for the delegate to a different host in the pool.

Syntax:

```
serviced pool set-conn-timeout POOLID TIMEOUT
```

The *TIMEOUT* value is specified with an integer followed by the units identifier. This command accepts the following units identifiers:

- `ms`, milliseconds
- `s`, seconds
- `m`, minutes
- `h`, hours

For example, `50ms` or `2m`.

serviced pool set-permission

The `serviced pool set-permission` command sets permission flags for hosts in the resource pool, *POOLID*. Before removing access to administrative functions (`--admin`), remove DFS access.

Syntax:

```
serviced pool set-permission [--dfs[=false]][--admin[=false]] POOLID
```

Command options:

--dfs[=false]]
Add or remove permission to mount the DFS. The default value is `true`.

--admin[=false]]

Add or remove permission to perform administrative functions DFS. The default value is `true`.

EXAMPLES

Give a resource pool distributed file system (DFS) access:

```
serviced pool set-permission --dfs pool_01_140620
```

Give a resource pool access to administrative functions:

```
serviced pool set-permission --admin pool_01_140620
```

Remove DFS access permission from a resource pool.

Note If you need to remove access to administrative functions, first remove DFS access.

```
serviced pool set-permission --dfs=false pool_01_140620
```

Remove resource pool access to administrative functions.

Note If you need to remove access to administrative functions, first remove DFS access.

```
serviced pool set-permission --admin=false pool_01_140620
```

Set the connection timeout value to 3 minutes:

```
serviced pool set-conn-timeout pool_01_140620 3m
```

serviced restore

The `serviced restore` command restores an instance of an application from a backup file on the same system. You can also duplicate your instance on a similar cluster for testing or failover purposes by restoring a backup file to a new, similarly configured cluster.

If you are restoring from a backup that was taken on another system, copy the backup archive file to the target system.

USAGE

```
serviced restore [command options] [arguments...]
```

OPTIONS

--help, -h

Show the help for an option.

serviced script

The `serviced script` command verifies or performs the commands in a script file.

USAGE

A script file is a text file that contains commands to automate common or repetitive tasks and tasks that might require specific services or conditions.

The `serviced script` command provides three subcommands.

`help`
: Display the help message.

`parse`
: Verify the syntax of a script file.

`run`
: Perform the commands in a script file.

The correct invocation of `serviced script run` depends on whether the `REQUIRE_SVC` command is present in a script file.

- If a script file does not include `REQUIRE_SVC`, no additional parameters are required. For example:

```
serviced script run task1.txt
```

- If a script file includes `REQUIRE_SVC`, the `--service` parameter is required. For example:

```
serviced script run task2.txt --service Zenoss.core
```

The log file of a `serviced script run` invocation is `/var/log/serviced/script-`*`TIMESTAMP`*`-`*`$USER`*`.log`

Note To commit a container, a `serviced script run` invocation must be performed on the Control Center master host.

SYNTAX

The script file syntax rules are as follows:

- Lines that contain no text and lines that start with the number sign character (#) are ignored.
- Lines are terminated with LF or CR+LF.
- A command and its arguments cannot span lines.
- The maximum number of characters per line (command and arguments) is 300000.
- Unless otherwise noted, all command arguments are treated as strings.

COMMANDS

Commands are performed in the order in which they occur in a script. Scripts terminate on completion and when a command returns an exit code other than zero.

`DESCRIPTION` ***argument...***
: A statement about the script.

  Scripts may contain one or zero `DESCRIPTION` commands. At least one argument is required.

`VERSION` ***argument***
: A revision reference for the script.

  Scripts may contain one or zero `VERSION` commands. Only one argument is supported.

`REQUIRE_SVC`

The script needs a reference service in order to perform some or all of its tasks. The service is specified with the `--service` parameter of the `serviced script run` command.

Scripts may contain one or zero `REQUIRE_SVC` commands.

`SNAPSHOT`

Perform a snapshot. If a script command fails, `serviced` rolls back to the most recent snapshot.

The `REQUIRE_SVC` command must be present in the script.

Scripts may contain multiple `SNAPSHOT` commands.

`SVC_USE` ***Image-ID***

Use the specified image for script commands that occur after this `SVC_USE` command. If your application uses multiple images, enter additional `SVC_USE` commands to specify each image. If the specified image is not present in the local Docker registry, `serviced` attempts to pull it from Docker Hub.

The `REQUIRE_SVC` command must be present in the script.

Scripts may contain multiple `SVC_USE` commands. Only one argument is supported.

`SVC_RUN` ***Service Run-Command arguments***

Invoke one of the pre-defined commands associated with a service.

Service must be the absolute path of a service, with each service in the path separated by the solidus character (`/`). For example, `Zenoss.core/Zope`.

The `REQUIRE_SVC` command must be present in the script.

Scripts may contain multiple `SVC_RUN` commands. Multiple arguments are supported.

`SVC_EXEC [`**COMMIT** | **NO_COMMIT**`]` ***Service argument...***

Start a new container to run arbitrary commands. (Equivalent to a non-interactive invocation of `serviced service shell`.)

When `COMMIT` is specified, changes are committed on successful completion of the commands in *argument*. When `NO_COMMIT` is specified, changes are not committed.

Service must be the absolute path of a service, with each service in the path separated by the solidus character (`/`). For example, `Zenoss.core/Zope`.

The `REQUIRE_SVC` command must be present in the script.

Scripts may contain multiple `SVC_EXEC` commands.

`SVC_START {`**auto** | **recurse**`}` ***Service***

Start a new instance of *Service*.

If `auto` or `recurse` is not specified, all configured instances of *Service* are started. If `auto` or `recurse` is specified, all configured instances of *Service* and all of their child services are started.

Service must be the absolute path of a service, with each service in the path separated by the solidus character (`/`). For example, `Zenoss.core/Zope`.

The `REQUIRE_SVC` command must be present in the script.

Scripts may contain multiple `SVC_START` commands.

`SVC_STOP {`**auto** | **recurse**`}` ***Service***

Stop the specified service.

If `auto` or `recurse` is not specified, all instances of *Service* are stopped. If `auto` or `recurse` is specified, all instances of *Service* and all of their child services are stopped.

Service must be the absolute path of a service, with each service in the path separated by the solidus character (`/`). For example, `Zenoss.core/Zope`.

The `REQUIRE_SVC` command must be present in the script.

Scripts may contain multiple `SVC_STOP` commands.

`SVC_RESTART {`**auto** | **recurse**`}` ***Service***

Restart the specified service.

If `auto` or `recurse` is not specified, all instances of *Service* are restarted. If `auto` or `recurse` is specified, all instances of *Service* and all of their child services are restarted.

Service must be the absolute path of a service, with each service in the path separated by the solidus character (`/`). For example, `Zenoss.core/Zope`.

The `REQUIRE_SVC` command must be present in the script.

Scripts may contain multiple `SVC_RESTART` commands.

`SVC_WAIT` ***Service...*** `[`**started** | **stopped** | **paused**`]` ***Duration***

Pause *Duration* seconds, or pause until the specified service or services reach the `started`, `stopped`, or `paused` state. If the state is not reached when *Duration* expires, the command fails.

Duration must be an integer.

Each *Service* must be the absolute path of a service, with each service in the path separated by the solidus character (`/`). For example, `Zenoss.core/Zope`.

The `REQUIRE_SVC` command must be present in the script.

Scripts may contain multiple `SVC_WAIT` commands.

serviced service

The `serviced service` command lets you manage an application's individual services.

Use this command to perform administrative actions on a specific service or view service status information.

USAGE

```
serviced service [global options] command [command options] \
  [arguments...]
```

COMMANDS

The following commands are available for `serviced service`:

list

List all services.

status

Display the status of deployed services.

The status `emergency-stopped` indicates that Control Center performed an emergency shutdown of the service due to low storage. Before you can restart the service, resolve the space issue, and then use the `clear-emergency` command (listed below) to remove the emergency-shutdown flags. For more information, see *Emergency shutdown of services* on page 21.

add

Add a service.

clear-emergency

Reset the emergency-shutdown flag for a service that was shut down due to a low-storage condition.

clone

Clone a service.

remove, rm
Remove a service.

edit
Edit a service in a text editor.

assign-ip
Assign an IP address to service endpoints that require an explicit IP address.

start
Start one or more services.

restart
Restart one or more services.

stop
Stop one or more services.

shell
Start a service instance.

run
Run a service command in a service instance.

attach
Run an arbitrary command in a running service container.

action
Run a predefined action in a running service container.

logs
Display the log contents for a running service container by calling `docker logs`.

list-snapshots
List all snapshots of a service.

snapshot
Take a snapshot of a service.

endpoints
List the endpoints that are defined for a service.

public-endpoints
Manage public endpoints for a service.

help, h
Show a list of commands or the help for a single command.

GLOBAL OPTIONS

--generate-bash-completion

--help, -h
Shows the help for an option.

serviced service [start | stop]

By default, Control Center schedules services in the background to start, and stop. The asynchronous scheduling improves the speed of these operations, especially in large-scale installations. However, if you use a script that depends on synchronous scheduling that was used in earlier versions of Control Center, specify the command line option `--sync`.

Syntax:

```
serviced service [start | stop] [-s|--sync] ServiceID
```

Command option:

--sync, -s

Schedules services synchronously. Specify this flag if a script expects the `serviced service [start | stop]` command to wait to return until the service operation has been scheduled. If this flag is not specified, services are scheduled asynchronously, in the background.

serviced service restart

By default, Control Center schedules services in the background to restart to improve the speed of the operation, especially in large-scale installations. However, if you use a script that depends on synchronous scheduling that was used in earlier versions of Control Center, specify the command line option `--sync`.

Syntax:

```
serviced service restart [commandOptions] \
    {ServiceID | instanceID}
```

Command option:

--auto-launch

Recursively schedules child services.

--sync, -s

Schedules services synchronously. Specify this flag if a script expects the `serviced service restart` command to wait to return until the service operation has been scheduled. If this flag is not specified, services are scheduled asynchronously, in the background.

--rebalance

Stops all instances of a service before restarting, instead of performing a serial restart of multi-instance services.

serviced snapshot

The `serviced snapshot` command administers environment snapshots.

Use this command to create a short-term save point of a system.

USAGE

```
serviced snapshot [global options] \
  command [command options] [arguments...]
```

COMMANDS

The following commands are available for `serviced snapshot`:

list

List all snapshots.

add
Take a snapshot of an existing service.

remove, rm
Remove an existing snapshot.

commit
Take a snapshot of and commit a given service instance.

rollback
Restore the environment to the state of the given snapshot.

tag
Tag an existing snapshot with TAG-NAME.

untag
Remove a tag from an existing snapshot.

help, h
Show a list of commands or the help for a single command.

OPTIONS

--generate-bash-completion

--help, -h
Show the help for an option.

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

Create thin pools on separate block devices:

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

Create thin pools on separate partitions:

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

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

```
lvextend -L+300G zenoss/serviced-pool
serviced-storage -o dm.thinpooldev=/dev/mapper/zenoss-serviced--pool \
```

```
  resize -d /opt/serviced/var/volumes 58uuetj38draeu9alp6002b1y 200G
```

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

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

Control Center configuration 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`

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 (`,`).

Glossary

application
: A collection of one or more software programs that have been converted into Docker containers.

cluster
: The collection of hosts in one or more Control Center resource pools.

delegate host
: A host that runs the application services scheduled for the resource pool to which it belongs. A system can be configured as delegate or master.

master host
: The host that runs the application services scheduler, the Docker registry, the distributed file system, and other internal services, including the server for the Control Center browser interface. A system can be configured as delegate or master. Only one system in a Control Center cluster can be the master.

resource pool
: A collection of one or more hosts, each with its own compute, network, and storage resources. The name of the default resource pool is `default`. Control Center uses resource pools (except the default pool) to run services on specific hosts.

service
: A process and its supporting files that Control Center runs in a single container to provide specific functionality as part of an application.

service definition
: A service definition contains the information that Control Center needs to start and manage a service, in JavaScript Object Notation (JSON) format.

service template
: A service template contains one or more service definitions, in JavaScript Object Notation (JSON) format.

serviced
: The name of the Control Center service and a command-line client for interacting with the service.

tenant
: An application that Control Center manages.