Zenoss Control Center Help

Release 1.1.x

Zenoss, Inc.

www.zenoss.com

Zenoss Control Center Help

Part Number: 1080.15.223

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

Contents

Administration reference 1

This chapter contains several procedures for managing Control Center clusters.

Reducing high load averages on AWS hosts

When Docker is using the `devicemapper` storage driver, a Control Center instance on an Amazon Web Services host may experience persistent, high averages for CPU load.

To diagnose and remedy this condition, perform this procedure.

1 Log in to the master host or a resource pool host as a user with `sudo` and `docker` privileges.
2 Display information about Docker.

```
sudo docker info | grep -A 2 'Storage Driver:'
```

Sample output:

```
Storage Driver: devicemapper
 Root Dir: /var/lib/docker/devicemapper
 Dirs: 21
```

- If the storage driver is `devicemapper`, continue this procedure.
- If the storage driver is not `devicemapper`, stop this procedure The cause of high load averages lies elsewhere.

3 Stop the `serviced` and `docker` daemons.

```
sudo stop serviced && sudo stop docker
```

4 Create a variable for the Docker defaults file on your platform.

- RHEL/CentOS hosts: `DEFAULTS=/etc/sysconfig/docker`
- Ubuntu hosts: `DEFAULTS=/etc/default/docker`

5 Edit the Docker defaults file.
The DOCKER_OPTS variable specifies startup options for the `docker` daemon.

```
grep '^[^#]*DOCKER_OPTS' $DEFAULTS
```

- If the preceding command returns a result, add the required option to the existing options.

```
reqopt='--storage-opt dm.blkdiscard=false'
opt=$(grep '^[^#]*DOCKER_OPTS' $DEFAULTS)
opt=${opt#*=}; optqc=${opt:0:1}; optlen=${#opt}
opt=${opt:1:($optlen-2)}
MYOPT="${optqc}${opt} ${reqopt}${optqc}"
sudo sed -i -e \
  's|^[^#]*DOCKER_OPTS=.*$|DOCKER_OPTS='"${MYOPT}"'|' $DEFAULTS
```

- If the preceding command does not return a result, define the DOCKER_OPTS variable with the required option.

```
REQOPT='\nDOCKER_OPTS="--storage-opt dm.blkdiscard=false"'
sudo sed -i -e '/^#DOCKER_OPTS=/ s|$|'"${REQOPT}"'|' $DEFAULTS
```

6 Start the `docker` and `serviced` daemons.

```
sudo start docker &&  sudo start serviced
```

Converting a single-host deployment to multi-host

You may add hosts to a single-host deployment by following the steps in this procedure.

1 Install Control Center on resource pool hosts.
2 Log in to the Control Center web interface, and then stop all applications.
3 Log in to the master host.

- For Ubuntu hosts, log in as a user with `sudo` and `docker` privileges.
- For RHEL or CentOS hosts, log in as `root`.

4 Stop the `serviced` daemon.

- Ubuntu: `sudo stop serviced`
- RHEL/CentOS: `systemctl stop serviced`

5 Configure the master host for multi-host deployment.

To enable a multi-host deployment, uncomment and change the following variables.

SERVICED_REGISTRY

Determines whether `serviced` uses a local registry to store Docker images. Set the value to 1, true.

SERVICED_AGENT

Determines whether a `serviced` instance performs agent tasks. Agents run application services scheduled for the resource pool to which they belong. Set the value to 1, true.

SERVICED_MASTER

Determines whether a `serviced` instance performs master tasks. The master runs the application services scheduler and other internal services, including the server for the Control Center web interface. A `serviced` instance may be configured as both an agent and a master. Set the value to 1, true.

The following commands make the required edits to `/etc/default/serviced`.

```
EXT=$(date +"%j-%H%M%S")
sudo sed -i.${EXT} -e 's|^#[^S]*\(SERVICED_REGISTRY=\).|\11|' \
  -e 's|^#[^S]*\(SERVICED_AGENT=\).|\11|' \
  -e 's|^#[^S]*\(SERVICED_MASTER=\).|\11|' \
   /etc/default/serviced
```

6 Start `serviced`.

- Ubuntu: `sudo start serviced`
- RHEL/CentOS: `systemctl start serviced`

7 Synchronize the local Docker registry with the resource pool hosts.

- Ubuntu: `sudo serviced docker sync`
- RHEL/CentOS: `serviced docker sync`

8 In the Control Center web interface, start applications.

Deploying multiple instances of a service

Control Center can not deploy more than one instance of an application that includes virtual host names, unless the virtual host names are unique. Follow this procedure to modify an application service template and deploy multiple instances of an application.

1 Log in to the Control Center master host as a user with read and write privileges in `/opt/serviced/templates`.

- For Ubuntu hosts, log in as a user with `sudo` privileges.
- For RHEL or CentOS hosts, log in as `root`.

2 Change directory to the Control Center templates directory, and copy the template file to modify.

```
cd /opt/serviced/templates
sudo cp App-Template.json App-Template-Copy.json
```

3 Use a text editor to make the following changes to the new service template file:

a Optional: Change the value of the `Name` field on line 3.

This change differentiates application service templates.

b Change the values of the defined `VHosts` fields in all subservice `Endpoints` objects.

4 Use `serviced` to add the modified template.

```
sudo serviced template add App-Template-Copy.json
```

5 Log in to the Control Center web interface, and deploy the new application.

Changing the name or IP address of the master host

If you change the hostname or IP address of the Control Center master host, the Zenoss applications that you manage through Control Center cannot run until they are made aware of the master host's new system information. Use this procedure to update the master host system information in the Control Center so that managed applications can run properly.

1 Log in to the Control Center web interface.

2 In the **Actions** column of the **Applications** table, click **Stop** for each Zenoss application running on the master host.

3 In the **Stop Services** dialog, click **Stop Services**.

4 Log in to the Control Center master host as a user with `sudo` privileges.

5 Remove the `zookeeper` file:
`rm -rf /opt/serviced/var/isvcs/zookeeper`

6 Start the `serviced` service:

- Ubuntu: `sudo start serviced`

- RHEL/CentOS: `systemctl start serviced`

7 Return to the Control Center web interface.

8 On the **Hosts** page, locate the master host with the old system information and click **Delete**.

9 In the **Remove Host** dialog, click **Remove Host**.

   The master host with the old information is removed from the **Hosts** table.

10 Click **Add Host**.

11 In the **Add Host** dialog, specify the new **Host and port**, the **Resource Pool ID**, and **RAM Committment**, and then click **Add Host**.

   The master host is added to the **Hosts** table with the new system information.

12 On the **Applications** page, click **Start** to restart all applications.

Using the Control Center command-line interface 2

The Control Center command-line interface (CLI) and the Control Center web interface both use the same application programming interface. Any action performed in the CLI can be performed in the web interface.

Invoking `serviced`

To use the CLI, you need a Linux shell account on hosts in Control Center pools. The account needs to be a member of the `docker` group.

To invoke `serviced`, you simply log into the Control Center master host and issue a `serviced` command. For example, to list the services running on the master host, you enter the following command while logged on to that host:

```
serviced service list
```

Currently, if you attempt to invoke `serviced` on a host other than the Control Center master host, you must specify the master host and port manually, using the `endpoint` global option.

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

Replace *Master-Host* with the hostname or IP address of the Control Center master host.

Note To commit a container, a `serviced` CLI session must be run 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*.

zenoss

GLOBAL OPTIONS

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

The local Docker registry to use. The value of *Master-Hostname* is the hostname of the master host, not its IP address.

--static-ip *IP-Address* **[--static-ip** *IP-Address***] ...**

One or more static IP addresses for a `serviced` instance to advertise. Static IP addresses are declared in service definition templates, but must be passed to an instance with this option.

--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. NOTE: For release 1.0, only port 4979 is supported.

--outbound *IP-Address*

The outbound IP address.

--uiport : *Port*

The port on which this `serviced` instance listens for HTTPS requests. The default value is 443, unless *SERVICED_UI_PORT* is set in the configuration file.

--listen : *Port*

The `serviced` RPC endpoint on the local host. The default value of *Port* is 4979. NOTE: For release 1.0, only port 4979 is supported.

--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 multiplexer port. The default value is 22250, unless *SERVICED_MUX_PORT* is set in the configuration file.

--var *Path*

The location in which `serviced` stores application data. The default value is *`$SERVICED_HOME`*`/var`, unless *SERVICED_VARPATH* is set in the configuration file.

--keyfile *Path*

The path of a TLS key file. The default value is *`$TMPDIR`*`/zenoss_key.[0-9]+`, unless *SERVICED_KEY_FILE* is set in the configuration file.

--certfile *Path*

The path of a TLS certificate file. The default value is *`$TMPDIR`*`/zenoss_cert.[0-9]+`, unless *SERVICED_CERT_FILE* is set in the configuration file.

--zk *Host***:** *Port* **[--zk** *Host***:** *Port***] ...**

One or more ZooKeeper endpoints. The value of *Host* is the hostname or IP address of the ZooKeeper host. The default value of *Port* is 2181. If multiple endpoints are specified, `serviced` will try each in turn until it connects to a working server.

--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 for the underlying file system where application data is stored (by default, `/opt/serviced/var`). The supported drivers are `rsync` and `btrfs`. The default value is `rsync`, unless *SERVICED_FS_TYPE* is set in the configuration file.

--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 to wait before removing a stopped container. The default value is 86400 seconds (24 hours), unless *SERVICED_MAX_CONTAINER_AGE* is set in the configuration file.

--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 16-bit private subnet to use for virtual IPv4 addresses. The default value is 10.3, unless *SERVICED_VIRTUAL_ADDRESS_SUBNET* is set in the configuration file.

--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` web interface. On Ubuntu systems, the default value of *Group* is `sudo`; on RHEL/CentOS systems, the default is `wheel`. If *SERVICED_VIRTUAL_ADDRESS_SUBNET* is set in the configuration file, its value is used instead of the default.

--report-stats

Enable reporting statistics in a container.

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

The endpoint of the OpenTSDB reader daemon, for `serviced` statistics. 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.

--stats-period *Duration*

The number of seconds to wait between polls (health checks) of hosts in resource pools. The default value of *Duration* is 10, unless *SERVICED_STATS_PERIOD* is set in the configuration file.

--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***] ...**

Arguments to pass to internal services. The default value is no arguments, unless *SERVICED_ISVCS_ENV_[0-9]+* is set in the configuration file.

--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`.

--max-rpc-clients *Count*

The maximum number of clients the `serviced` RPC endpoint supports. The default value is 3, unless *SERVICED_MAX_RPC_CLIENTS* is set in the configuration file.

--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.

--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 *logstash* endpoint to which log data are sent. The default value of *Host* is the IP address of the 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.

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

The administrative endpoint of the *logstash* 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 data in the logstash server before purging it. The default value of *Duration* is 1, 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.

--logstash-max-size *Quantity*

The maximum amount of logstash data to keep, in gigabytes. The default value of *Quantity* is 10, unless *SERVICED_LOGSTASH_MAX_SIZE* is set in the configuration file. When this argument and `--logstash-max-days` are used at the same time, both conditions are evaluated and enforced.

--v *Level*

The relative amount of INFO messages `serviced` writes to the system log. The value of *Level* may be 0 (minimum) to 5 (maximum). The default value is 0, unless *SERVICED_LOG_LEVEL* is set in the configuration file.

--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.

--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 and services into a `tar` archive file.

`docker`

Administer Docker.

`[help|h]`
: Display a global or command-specific help message.

`host`
: Administer hosts.

`log`
: Administer logs.

`metric`
: Administer metrics.

`pool`
: Administer resource pools.

`restore`
: Reconstruct templates and services from a a `tar` archive file created with `backup`.

`script`
: Administer services.

`service`
: Administer services.

`snapshot`
: Administer snapshots.

`template`
: Administer templates.

`version`
: Display `serviced` version information.

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.

- RHEL/CentOS: `sudo systemctl [start|stop|reload] serviced`
- Ubuntu: `sudo [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 an agent host, 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.

ENVIRONMENT

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

FILES

`/etc/default/serviced`

serviced service

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

USAGE

Use this command to perform administrative actions on specific service.

SYNTAX

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

COMMANDS

The following commands are available for `serviced service`:

list

List all services.

status

Displays the status of deployed services.

add

Adds a new service.

clone

Clones a new service.

migrate, mig

Migrates an existing service.

remove, rm

Removes an existing service.

edit

Edits an existing service in a text editor.

assign-ip

Assigns an IP address to service endpoints that require an explicit IP address.

start

Starts a service.

restart

Restarts a service.

stop

Stops a service.

proxy

Starts a server proxy for a container.

shell

Starts a service instance.

run

Runs 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

Displays the log contents for a running service container by calling Docker logs.

list-snapshots

Lists all snapshots for a service.

snapshot

Takes a snapshot of the service.

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 service clone`

Creates a clone of an existing service.

USAGE

Use this command to quickly create a clone of an existing service. The command returns the cloned service's new ID.

Use `serviced service list` to view the cloned service.

The cloned service is not automatically started. To start the service use `serviced service start`.

SYNTAX

```
serviced service clone --suffix { SERVICEID | SERVICENAME |
[POOL/]...PARENTNAME.../SERVICENAME }
```

OPTIONS

--suffix

A name that is appended to the service, volumes, and endpoints of the newly cloned service.

If a suffix is not specified, a dash and the first 12 characters of the `SERVICEID` is appended to the name, for example: `-b27482e7a1b6`

SERVICEID

A unique string of characters that identify the service. (How many chars long?)

SERVICENAME

The name of the service.

POOL/...PARENTNAME.../SERVICENAME

The path to the service, starting with the Resource Pool name, parent name, and finally the service name.

EXAMPLE

The following example command creates a clone of the `mariadb-model` service and appends it with the suffix "-AusHub":

```
root@zenoss:~# serviced service clone --suffix=-AusHub mariadb-model

anm4tpiqxj1gih6puzl7apn6i
```

The following example command lists all instances of the `mariadb-model` service:

```
root@zenoss:~# serviced service list | grep mariadb-model

mariadb-model 7h7uefodmvwrx81p31sp46ss7 1 .../b034bfj.../resmgr_5.0 default
1 auto waterloo mariadb-model-AusHub anm4tpiqxj1gih6puzl7apn6i 1 .../
b034bfj.../resmgr_5.0 default 0 auto waterloo ip-10-111-23-218 Y
```

The following example command starts the cloned service:

```
root@zenoss:~# serviced service start mariadb-model-AusHub

Scheduled 1 service(s) to start
```

`/etc/default/serviced`

Control Center (`serviced`) is a single binary that behaves differently based on the environment variables defined in its startup script.

- On Ubuntu systems, the startup script is `/etc/init/serviced.conf`.
- On RHEL/CentOS systems the startup script is `/etc/systemd/system/multi-user.target.wants/serviced.service`.

The startup script reads the `serviced` defaults file, `/etc/default/serviced`. The startup script includes a few environment variables, but the values defined in the defaults file take precedence over the values defined in the script.

The following list describes the contents of `/etc/default/serviced` and their effects on the `serviced` daemon.

`HOME`

Default: `/root`

The path `docker` uses to locate the `.dockercfg` authentication file. Docker Hub credentials are stored in the file.

`GOMAXPROCS`

Default: 2

The maximum number of CPU cores `serviced` uses.

`SERVICED_AGENT`

Default: `1` (true)

Determines whether a `serviced` instance performs agent tasks. Agents run application services scheduled for the resource pool to which they belong. The `serviced` instance configured as the master runs the scheduler. A `serviced` instance may be configured as agent and master, or just agent, or just master.

`SERVICED_MASTER`

Default: `1` (true)

Determines whether a `serviced` instance performs master tasks. The master runs the application services scheduler and other internal services, including the server for the Control Center web interface. A `serviced` instance may be configured as agent and master, or just agent, or just master. Only one `serviced` instance in a Control Center cluster may be the master.

`SERVICED_MASTER_POOLID`

Default: `default`

The name of the resource pool to which the `serviced` instance configured as master belongs.

`SERVICED_ZK`

Default: `{{SERVICED_MASTER_IP}}:2181`

The list of endpoints in `serviced`'s *ZooKeeper* ensemble, separated by the comma character (`,`). Currently, the `serviced` instance configured as master is the only host in the ZooKeeper ensemble. Replace `{{SERVICED_MASTER_IP}}` with the IP address of the host on which the `serviced` instance configured as master is installed.

`SERVICED_DOCKER_REGISTRY`

Default: `{{SERVICED_MASTER_IP}}:5000`

The IP address and port number of the `serviced` registry host. Replace `{{SERVICED_MASTER_IP}}` with the IP address of the host on which the `serviced` instance configured as master is installed, or an alternate host.

`SERVICED_OUTBOUND_IP`

Default: `10.0.0.29`

The outbound IP address to set when Control Center is used in a private network.

Note Setting the Docker *HTTP_PROXY* or *HTTPS_PROXY* environment variables prevents access to the `serviced` outbound IP address. To enable access, unset the variables, and then reboot the host.

`SERVICED_STATIC_IPS`

Default: (none)

One or more static IP addresses reserved for `serviced` to use when it advertises external IP addresses on behalf of services it manages, separated by the comma character (`,`).

`SERVICED_ENDPOINT`

Default: `{{SERVICED_MASTER_IP}}:4979`

The IP address and port number of the `serviced` RPC endpoint on the master host. Replace `{{SERVICED_MASTER_IP}}` with the IP address of the host on which the `serviced` instance configured as master is installed.

`SERVICED_MAX_RPC_CLIENTS`

Default: `3`

The maximum number of clients the `serviced` RPC endpoint supports. 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.

`SERVICED_RPC_PORT`

Default: `4979`

The port on which `serviced` listens for RPC requests. NOTE: For release 1.0, only port 4979 is supported.

`SERVICED_UI_PORT`

Default: `443`

The port on which the master `serviced` instance listens for HTTPS requests. (The web interface port.)

`SERVICED_MUX_PORT`

Default: `22250`

The port `serviced` uses for multiplexing on its private subnet.

SERVICED_VARPATH

Default: `/opt/serviced/var`

The location in which `serviced` stores application data.

SERVICED_KEY_FILE

Default: `$TMPDIR/zenoss_key.[0-9]+`

The path of a TLS key file. The default, insecure key file is created when the `serviced` web server first starts, from a public key that is compiled into `serviced`.

SERVICED_CERT_FILE

Default: `$TMPDIR/zenoss_cert.[0-9]+`

The path of a TLS certificate file. The default, insecure certificate file is created when the `serviced` web server first starts, from a public certificate that is compiled into `serviced`.

Note Currently, certificates with passphrases are not supported.

SERVICED_FS_TYPE

Default: `btrfs`

The driver for the underlying file system where application data is stored (by default, `/opt/serviced/var`). The supported types are `rsync`, `btrfs`, and `devicemapper`.

Note Only `btrfs` and `devicemapper` are supported in production deployments. The `rsync` type is appropriate only in development deployments.

SERVICED_VHOST_ALIASES

Default: `foobar.com,example.com`

The list of virtual host aliases to use in virtual host multiplexing, separated by the comma character (`,`).

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`

The 16-bit private subnet to use for `serviced`'s virtual IPv4 addresses. RFC 1918 restricts private networks to the 10.0/24, 172.16/20, and 192.168/16 address spaces. However, `serviced` accepts any valid, 16-bit, IPv4 address space for its private network.

Note This value affects the values of SERVICED_STATIC_IPS.

SERVICED_LOG_LEVEL

Default: `0`

The relative amount of INFO messages `serviced` writes to the system log. The values may be 0 (minimum) to 5 (maximum). On Ubuntu systems, the log file is `/var/log/upstart/serviced.log`. On RHEL/CentOS systems, the log file is `/var/log/journal/serviced.log`, if persistent storage for log files is enabled. Otherwise, log data is stored only temporarily by the `systemd` journal.

SERVICED_LOG_ADDRESS

Default: `{{SERVICED_MASTER_IP}}:5042`

The *logstash* endpoint to which log data are sent. Replace `{{SERVICED_MASTER_IP}}` with the IP address of the host on which the `serviced` instance configured as master is installed.

SERVICED_LOGSTASH_ES

Default: `{{SERVICED_MASTER_IP}}:9100`

The endpoint of the *logstash* service. Replace `{{SERVICED_MASTER_IP}}` with the IP address of the host on which the `serviced` instance configured as master is installed.

`SERVICED_LOGSTASH_MAX_DAYS`

Default: `1`

The maximum number of days to keep data in the logstash server before purging it. When both this variable and *SERVICED_LOGSTASH_MAX_SIZE* are set, both conditions are evaluated and enforced.

`SERVICED_STATS_PORT`

Default: `{{SERVICED_MASTER_IP}}:8443`

The endpoint of the OpenTSDB reader daemon, for `serviced` statistics. Replace `{{SERVICED_MASTER_IP}}` with the IP address of the host on which the `serviced` instance configured as master is installed.

`SERVICED_STATS_PERIOD`

Default: `10`

The number of seconds to wait between polls (health checks) of hosts in resource pools.

`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: (empty)

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*:

```
sudo docker ps | awk '/isvcs:/{print substr($NF, \
  1, match($NF, /(-[0-9a-f]*){5}/) - 1 )}'
```

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=elasticsearch-logstash:ES_JAVA_OPTS=-Xmx4g
```

`SERVICED_ADMIN_GROUP`

Default: `sudo` (Ubuntu) `wheel` (RHEL/CentOS)

The name of the Linux group on the Control Center master host whose members are authorized to use the Control Center web 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 `root`, or members of the `sudo` or `wheel` groups, may log in to the Control Center web interface.

`SERVICED_LOGSTASH_MAX_SIZE`

Default: `10`

The maximum amount of logstash data to keep, in gigabytes. When both this variable and *SERVICED_LOGSTASH_MAX_DAYS* are set, both conditions are evaluated and enforced.

`SERVICED_IPTABLES_MAX_CONNECTIONS`

Default: `655360`

The default value of this variable ensures that a `serviced` instance that is configured as an agent will not run out of connections if the `serviced` instance that is configured as master goes down. In testing this scenario, the connection count on remote hosts that are running a nominal number of collectors does not exceed 10,000 closed connections. The connections are automatically cleaned up by the kernel soon after the master host is back up.

`SERVICED_MONITOR_DFS_REMOTE_UPDATE_INTERVAL`

Default: `60` (1 minute)

The interval at which a `serviced` instance configured as agent modifies its `/opt/serviced/var/monitor/`*`IP-Address`* file.

`SERVICED_MONITOR_DFS_MASTER_INTERVAL`

Default: `180` (3 minutes)

The interval at which a `serviced` instance configured as master checks the modification times of the `/opt/serviced/var/monitor/`*`IP-Address`* files of active `serviced` instances configured as agents. The value of this variable must be a minimum of twice the value of the *SERVICED_MONITOR_DFS_REMOTE_UPDATE_INTERVAL* variable.

`SERVICED_MONITOR_DFS_MASTER_RESTART`

Default: `1` (true)

Determines whether the `serviced` master restarts NFS when it detects active agents that are out of synchronization.

`SERVICED_SERVICE_MIGRATION_TAG`

Default: `1.0.2`

Overrides the default value for the service migration image.

`SERVICED_SNAPSHOT_TTL`

Default: `12`

The number of hours a snapshot is retained before removal. To disable snapshot removal, set the value to 0.

`SERVICED_OPTS`

Default: (empty)

Arbitrary options for the `serviced` startup command.

`serviced script`

The `serviced script` command verifies or performs the commands in a script file. A script file is a text file that contains commands to automate common or repetitive tasks, and tasks that may require specific services or conditions.

Usage

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 invocaton 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.

Script file syntax

- 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 may not 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.

Example

```
# quilt.txt -- add Quilt to a Zenoss image
# Example invocation: serviced script run quilt.txt --service Zenoss.core
# On success, restart all services based on the zenoss-* image

DESCRIPTION Quilt Installation
VERSION zenoss-quilt-1.0
REQUIRE_SVC
SNAPSHOT

# Download the EPEL RPM
SVC_EXEC COMMIT Zenoss.resmgr yum install -y epel-release
# Download repository metadata
SVC_EXEC COMMIT Zenoss.resmgr yum makecache -y
# Install quilt
SVC_EXEC COMMIT Zenoss.resmgr yum install -y quilt
# Remove EPEL
SVC_EXEC COMMIT Zenoss.resmgr yum erase -y epel-release
# Clean up yum caches
SVC_EXEC COMMIT Zenoss.resmgr yum clean all
```

Glossary

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.