OPENSHIFT®
by Red Hat®

# OpenShift Enterprise 1 Administration Guide

A Guide to OpenShift Enterprise Operation and Administration

PressGang CCMS Build System

# OpenShift Enterprise 1 Administration Guide

## A Guide to OpenShift Enterprise Operation and Administration

PressGang CCMS Build System
Red Hat Engineering Content Services

## Legal Notice

## Keywords

## Abstract

This document provides information for operating and administering an OpenShift Enterprise installation.

# Table of Contents

# Preface

## 1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

### 1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

**`Mono-spaced Bold`**

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

**Proportional Bold**

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** →

> **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

***Mono-spaced Bold Italic*** or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

### 1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

### 1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

**Note**

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

**Important**

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

**Warning**

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

## 2. Getting Help

### 2.1. Do You Need Help?

If you experience difficulty with a procedure or other information described in this documentation, visit the Red Hat Customer Portal at http://access.redhat.com where you can:

- search or browse through a knowledgebase of technical support articles about Red Hat products
- submit a support case to Red Hat Global Support Services (GSS)
- access other product documentation

You can also access the OpenShift web site at https://openshift.redhat.com/ to find blogs, FAQs, forums, and other sources of information.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at https://www.redhat.com/mailman/listinfo. Click the name of any mailing list to subscribe to that list or to access the list archives.

### 2.2. We Need Feedback!

If you find a typographical or any other error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: http://bugzilla.redhat.com/ against the product **OpenShift Enterprise.**

When submitting a bug report, be sure to mention the manual's identifier: *Administration_Guide*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

# Chapter 1. Introduction

OpenShift Enterprise by Red Hat is a Platform-as-a-Service (PaaS) that enables developers to build and deploy web applications. OpenShift Enterprise provides a wide selection of programming languages and frameworks including Java, Ruby, and PHP. It also provides integrated developer tools to support the application lifecycle, including Eclipse integration, JBoss Developer Studio, and Jenkins. OpenShift Enterprise uses an open source ecosystem to provide a platform for mobile applications, database services, and more.

OpenShift Enterprise is built on Red Hat Enterprise Linux, which provides a secure and scalable multi-tenant operating system to address the needs of enterprise-class applications as well as providing integrated application runtimes and libraries.

OpenShift Enterprise brings the OpenShift PaaS platform to customer data centers, enabling organizations to implement a private PaaS that meets security, privacy, compliance, and governance requirements.

This document describes how to administer an OpenShift Enterprise installation. It provides information about common tasks, managing resources and monitoring, and includes a command reference section. It is intended for experienced system administrators.

Report a bug

# Chapter 2. Accessing Administrative Information

Report a bug

## 2.1. Determining Gear Ownership

On a node host, list the contents of the **`/var/lib/openshift/.httpd.d/`** directory to view the operational directories for gears. These directories have the format ***`UUID_domain_appname`***. For example, the following command shows a gear with an application named **`chess`** in the domain **`games`**:

```
# ls /var/lib/openshift/.httpd.d/

c13aca229215491693202f6ffca1f84a_games_chess
```

Report a bug

## 2.2. Viewing Resource Utilization

Report a bug

### 2.2.1. By Node

On a node host, use standard tools like **`free`** and **`vmstat`** to report memory utilization. Note that tools such as **`vmstat`**, **`top`**, and **`uptime`** report the number of processes waiting to execute, and may be artificially inflated because cgroups restrict each gear to a specific time slice rather than time-sharing between processes. This restriction enables OpenShift Enterprise to provide consistent CPU availability for each gear, but it means that active processes on gears may wait while the CPU is allocated to a less active gear. As a result, reported load average may routinely be close to the number of active gears.

Run the **`oo-idler-stats`** command to return a status summary of all the gears on a node host:

```
# oo-idler-stats

1 running, 1 idled, 0 half-idled for a total 2 of 3 (66.67 %)
```

In this example, there are three gears on the node: one is running, one is idle, and one is stopped. The **`half-idled`** state is deprecated and always returns 0.

Report a bug

# Chapter 3. Administrative Tasks

Report a bug

## 3.1. Clearing Broker and Console Application Cache

In production mode, Rails caches certain values on the broker for faster retrieval. Clearing this cache allows the retrieval of updated values. For example, the first time MCollective retrieves the list of cartridges available on your nodes, the list is cached so that subsequent requests for this information are processed more quickly. If you install a new cartridge, it is unavailable to users until the cache is cleared and MCollective retrieves a new list of cartridges. The Management Console also caches these values which can be cleared to retrieve updated values as well.

This caching is performed in multiple components:

- Each node maintains a database of facts about that node, which includes a list of installed cartridges.
- Using MCollective, a broker queries a node's facts database for the list of cartridges and caches the node's response.
- Using the broker's REST API, the developer console queries the broker for the list of cartridges and caches the broker's response.

The cartride lists are updated automatically at the following intervals:

- The node's database is refreshed every minute.
- The broker's cache is refreshed every six hours.
- The console's cache is refreshed every five minutes.

It is possible to clear these caches manually so that they will be refreshed immediately. Clearing only the broker cache (the **`--clear`** option) updates results observed via Client Tools, while clearing the console cache (the **`--console`** option) updates results in the Client Tools and Management Console. The following procedures describe how to clear these caches.

**Procedure 3.1. To Clear the Broker Application Cache Only:**

- On each broker host in your OpenShift Enterprise installation, run the following command:

```
# oo-admin-broker-cache --clear
```

**Procedure 3.2. To Clear the Broker and Console Application Cache:**

- On each broker host in your OpenShift Enterprise installation, run the following command (the **`--console`** option implies **`--clear`**):

```
# oo-admin-broker-cache --console
```

Report a bug

## 3.2. Removing User Applications

You can remove a user's application by using **`oo-admin-ctl-app`** to stop then delete it.

**Warning**

This procedure deletes all the data for the selected application and cannot be reversed.

**Procedure 3.3. To remove a user application:**

1. Stop the application by running the following command on the broker host:

```
# oo-admin-ctl-app -l username -a appname -c stop
```

2. Delete the application:

```
# oo-admin-ctl-app -l username -a appname -c destroy

!!!! WARNING !!!! WARNING !!!! WARNING !!!!
You are about to destroy the appname application.

This is NOT reversible, all remote data for this application will be removed.
Do you want to destroy this application (y/n): y
Successfully destroyed application: appname
```

3. If the standard **`stop`** and **`destroy`** commands fail, you can **`force-stop`** and **`force-remove`** the application. The **`force-`** commands do not wait for the proper shutdown sequence, so should only be used after the standard commands fail:

```
# oo-admin-ctl-app -l username -a appname -c force-stop
# oo-admin-ctl-app -l username -a appname -c force-destroy
```

Report a bug

## 3.3. Removing User Data

When a former user's application and domain data are no longer required, it is recommended that you remove this data in order to prevent a new user with the same username inheriting it.

**Warning**

The following procedure removes all of a user's application data from the system and cannot be reversed.

**Procedure 3.4. To remove user data:**

1. Prevent the user from creating more gears by running the following command on the broker host:

```
# oo-admin-ctl-user -l username --setmaxgears 0
```

2. Retrieve the user's namespace and application names:

```
# oo-admin-ctl-domain -l username | egrep -i '^name:|^Namespace:'

Namespace: testdomain
name: app1
name: app2
```

3. Remove the user's applications by running the following commands for each application:

```
# oo-admin-ctl-app -l username -a app1 -c stop
# oo-admin-ctl-app -l username -a app1 -c destroy
```

4. Delete the user's domain.

```
# oo-admin-ctl-domain -l username -c delete -n testdomain
```

5. The user's application data is now removed and the user cannot create any new applications; the account is effectively deactivated.

   To reactivate the user's account, set their maxgears higher than 0:

```
# oo-admin-ctl-user -l username --setmaxgears 5
```

Report a bug

## 3.4. Banning IP Addresses That Overload Applications

If an application user accesses an application with excessive frequency, you can block that user by banning their IP address.

**Note**

The ban instituted by the following procedure applies to all gears on the node host, not just the over-accessed gear.

**Procedure 3.5. To Ban an IP Address:**

1. View a CNAME to the node host where the application's gear is by running the following command:

```
# dig appname-domain.example.com
```

2. On the node host identified in the previous step, check the application's apache logs for unusual activity. For example, a high frequency of accesses (3 to 5 per second) from the same IP address in the **access_log** file may indicate abuse:

```
# tail -f /var/lib/openshift/appUUID/appname/logs/*
```

3. Ban the offending IP addresses by placing them in iptables, running the following command for each address:

```
# iptables -A INPUT -s IP_address -j DROP
```

4. If you are using a configuration management system, configure it appropriately to ban the offending IP addresses. For non-managed configurations, save your new iptables rules:

```
# service iptables save
```

Report a bug

## 3.5. Recovering Failed Node Hosts

A node host that fails catastrophically can be recovered if the gear directory **`/var/lib/openshift`** has been stored in a fault-tolerant way and can be restored. SELinux contexts must be preserved with the gear directory in order for recovery to succeed.

In practice, this scenario occurs rarely, especially when node hosts are virtual machines in a fault-tolerant infrastructure rather than physical machines.

**Important**

Scaled applications cannot be recovered onto a node host that has a different IP address than the original node host.

**Procedure 3.6. Recovering a Failed Node Host**

1. Install a node host with the same hostname and IP address as the one that failed.
   a. The hostname DNS A record can be adjusted if the IP address must be different, but note that the application CNAME and database records all point to the hostname and cannot be easily changed.
   b. Ensure the **`mcollective`** service is not running on the new node host.

      ```
      # service mcollective stop
      ```

   c. Duplicate the old node host's configuration on the new node host, ensuring in particular that the gear profile is the same.
2. Mount **`/var/lib/openshift`** from the original, failed node host.
   a. Verify that the SELinux contexts stored on the **`/var/lib/openshift`** volume are preserved on the new node host.
3. Recreate **`/etc/passwd`** entries for all the gears using the following steps:
   a. Get the list of UUIDs from the directories in **`/var/lib/openshift`**.
   b. For each UUID, get the Unix UID and GID values corresponding to the group ID of the directory **`/var/lib/openshift/`*`UUID`***. See the fourth value in the output from the following command:

      ```
      # ls -l -n /var/lib/openshift/UUID
      ```

   c. Create the corresponding entries in **`/etc/passwd`**, using another node's **`/etc/passwd`** file for reference.
4. Reboot the new node host to activate all changes, start the gears, and allow MCollective and other services to run.

Report a bug

## 3.6. Managing Custom and Community Cartridges

In addition to cartridges provided and supported by Red Hat, you can use *custom* and *community cartridges*. The following table describes the cartridge types available to you and indicates their Red Hat support levels.

**Table 3.1. Cartridge Types**

| Type | Description | Red Hat Supported? |
|---|---|---|
| Standard cartridges | Shipped with OpenShift Enterprise | Yes, requires base OpenShift Enterprise entitlement |
| Premium cartridges | Shipped with OpenShift Enterprise | Yes, requires premium add-on OpenShift Enterprise entitlement |
| Custom cartridges | Created yourself either from scratch or based on another cartridge | No |
| Community cartridges | Community-contributed and made available either upstream in OpenShift Origin (see https://github.com/openshift/origin-community-cartridges) or elsewhere in the community | No |
| Partner cartridges | Developed by third-party partners | No, but may be supported directly by the partner |

**Note**

Red Hat supports the base OpenShift Enterprise platform on which custom and community cartridges run, but does not support or maintain the custom and community cartridges themselves. For more information about Red Hat's support for OpenShift Enterprise, refer to https://access.redhat.com/support/policy/updates/openshift/policies.html.

You can install and remove custom and community cartridges in OpenShift Enterprise. After new cartridges are installed, they appear as cartridge options for users in the Management Console and the Client Tools.

**Procedure 3.7. Installing Custom or Community Cartridges**

1. Run the **`oo-admin-cartridge`** command with the **`install`** action and specify the source directory of the custom or community cartridge to install:

```
# oo-admin-cartridge --action install --source /path/to/cartridge/

succeeded
```

2. Clear the broker and console caches to update the available cartridge list:

```
# oo-admin-broker-cache --console
```

**Procedure 3.8. Listing Installed Cartridges**

- Run the **`oo-admin-cartridge`** command with the **`--list`** option to list all installed cartridges. This command displays the vendor name, cartridge name, packaged software version, and cartridge version for each cartridge:

```
# oo-admin-cartridge --list

(redhat, jenkins-client, 1.4, 0.0.1)
(redhat, haproxy, 1.4, 0.0.1)
(redhat, jenkins, 1.4, 0.0.1)
(redhat, mock, 0.1, 0.0.1)
(redhat, tomcat, 8.0, 0.0.1)
(redhat, cron, 1.4, 0.0.1)
(redhat, php, 5.3, 0.0.1)
(myvendor, mycart, 1.1, 0.0.1)
(redhat, ruby, 1.9, 0.0.1)
(redhat, perl, 5.10, 0.0.1)
(redhat, diy, 0.1, 0.0.1)
(redhat, mysql, 5.1, 0.2.0)
```

**Procedure 3.9. Removing Cartridges**

1. Run the **`oo-admin-cartridge --list`** command to identify the cartridge name, packaged software version, and cartridge version of the cartridge you want to remove:

   ```
   # oo-admin-cartridge --list

   (redhat, jenkins-client, 1.4, 0.0.1)
   (redhat, haproxy, 1.4, 0.0.1)
   (redhat, jenkins, 1.4, 0.0.1)
   ...
   ```

2. Run the **`oo-admin-cartridge --action erase`** command with the cartridge information identified in the previous step:

   ```
   # oo-admin-cartridge --action erase --name mycart --version 1.1 --
   cartridge_version 0.0.1

   succeeded
   ```

3. Clear the broker and console caches to update the available cartridge list:

   ```
   # oo-admin-broker-cache --console
   ```

Report a bug

## 3.7. Applying Updates to OpenShift Enterprise

OpenShift Enterprise relies on a complex set of dependencies; to avoid problems, caution is required when applying software updates to broker and node hosts.

For bug fixes and other targeted changes, updated RPMs are released in existing channels. Please read errata advisories carefully before updating, as these provide instructions on how to apply updates safely and include details of required service restarts and configuration changes. For example, when updating **`rubygem`** packages required by the broker application, it is necessary to restart the openshift-broker service. This step regenerates the bundler **`Gemfile.lock`** file and allows the broker application and related administrative commands to use the updated gems.

For systemic version updates requiring formal migration scripts and lockstep package updates, refer to the latest OpenShift Enterprise *Release Notes* for upgrade steps.

Report a bug

# Chapter 4. Managing Resources

Report a bug

## 4.1. Setting Default Gear Quotas and Sizes

You can set default gear quotas and sizes for all users by entering the desired values in the broker configuration file.

You must list gear sizes as VALID_GEAR_SIZES in **`/etc/openshift/broker.conf`** in order to create districts and applications with those sizes.

**Procedure 4.1. To set default gear quotas and sizes:**

- Open **`/etc/openshift/broker.conf`** on the broker host and edit the following defaults as desired:

```
# Comma-separated list of valid gear sizes available anywhere in the
installation
VALID_GEAR_SIZES="small,medium"

# Default number of gears to assign to a new user
DEFAULT_MAX_GEARS="100"

# Default gear size for a new gear if not otherwise specified
DEFAULT_GEAR_SIZE="small"

# Default gear sizes (comma-separated) allowed to a new user
DEFAULT_GEAR_CAPABILITIES="small"
```

Note that if ***DEFAULT_GEAR_CAPABILITIES*** is not set, ***DEFAULT_GEAR_SIZE*** determines the only gear size new users are allowed to use without an administrator granting access to other sizes.

Report a bug

## 4.2. Setting Gear Quotas for Specific Users

Use the **`oo-admin-ctl-user`** command to set individual user gear parameters. You can limit the number of gears a user is allowed, and change the gear sizes to which a user has access.

**Procedure 4.2. To display gear information for a user:**

- Run the following command on the broker host, replacing ***demo*** with the relevant user name:

```
# oo-admin-ctl-user -l demo

User demo:
   consumed gears: 3
   max gears: 100
   gear sizes: small
```

**Procedure 4.3. To limit the number of gears a user is allowed:**

- Run the following command on the broker host, replacing ***demo*** with the relevant user name and ***101*** with the desired number of gears:

```
# oo-admin-ctl-user -l demo --setmaxgears 101

Setting max_gears to 101... Done.

User demo:
   consumed gears: 3
   max gears: 101
   gear sizes: small
```

**Procedure 4.4. To add a gear size for a user:**

- Run the following command on the broker host, replacing ***demo*** with the relevant user name and ***medium*** with the desired gear size:

```
# oo-admin-ctl-user -l demo --addgearsize medium

Adding gear size medium for user demo... Done.

User demo:
   consumed gears: 3
   max gears: 101
   gear sizes: small, medium
```

**Procedure 4.5. To remove a gear size from a user:**

- Run the following command on the broker host, replacing ***demo*** with the relevant user name and ***medium*** with the desired gear size:

```
# oo-admin-ctl-user -l demo --removegearsize medium

Removing gear size medium for user demo... Done.

User demo:
   consumed gears: 3
   max gears: 101
   gear sizes: small
```

Report a bug

## 4.3. Providing Custom Resources and Settings to Applications

You can provide information to users' applications that should not be contained in their applications or gears, such as variable hostnames, passwords, or environment-specific settings, by placing it on the node hosts in your OpenShift Enterprise installation. This feature also enables you to make packages and services available to gears.

For example, if a database connection string is necessary for accessing a remote database, put the string into a file such as **/etc/database.connection.conf** and copy the file onto all the node hosts. All gears can now access this file, provided that file permissions and SELinux contexts allow it.

Report a bug

## 4.4. Setting Gear Supplementary Groups

When the broker creates a gear, the gear is assigned a UNIX user UID and a matching group UID. You can assign additional groups to the gears on a node, enabling you to make group-owned files available to all the gears on the node.

Use the **GEAR_SUPL_GRPS** setting in **/etc/openshift/node.conf** to designate additional groups for the gears on that node. Note that you must create a group using standard system commands before you can add it to **GEAR_SUPL_GRPS**. Separate multiple groups with commas.

```
GEAR_SUPL_GRPS="my_group,another_group"
```

**Note**

As a security measure, you cannot use the **root** and **wheel** groups as values for **GEAR_SUPL_GRPS**.

Report a bug

## 4.5. Managing Districts

Report a bug

### 4.5.1. Introduction

Districts define a set of node hosts within which gears can be easily moved to load-balance the resource usage of those nodes. While not required for a basic OpenShift Enterprise installation, districts provide several administrative benefits and their use is recommended.

Districts allow a gear to maintain the same UUID (and related IP addresses, MCS levels and ports) across any node within the district, so that applications continue to function normally when moved between nodes on the same district. All nodes within a district have the same profile, meaning that all the gears on those nodes are the same size (for example small or medium). There is a hard limit of 6000 gears per district.

This means, for example, that developers who hard-code environment settings into their applications instead of using environment variables do not experience problems due to gear migrations between nodes. The application continues to function normally because exactly the same environment is reserved for the gear on every node in the district. This saves developers and administrators time and effort.

Report a bug

### 4.5.2. Gear Placement Algorithm

Districts and node hosts both have a configured capacity for gears. For a node host, the default values configured in **/etc/openshift/resource_limits.conf** are:

```
max_gears=100
max_active_gears=100
```

**max_gears** is the total gear capacity, including active, stopped and idled applications.
**max_active_gears** is the capacity for active gears only.

District capacity is configured in MongoDB. Gear status does not affect district capacity, as districts reserve resources; they do not account for actual usage. In a district JSON record, ***max_capacity*** indicates how many gears can be placed in the district, while ***available_capacity*** indicates how many gears can still be placed.

Districts have a hard limit of 6,000 gears, as resources for the entire district must be reserved on each member node host to ensure they are available when a gear is moved between nodes. This means that in a district with only one node host, the district is full and cannot take additional gears if that node host reaches 6,000 gears. For a district with more than one node host, it is best practice to consider the district full when each node has a number of gears equal to 6,000 divided by the number of nodes. For example, the gear balancing algorithm keeps two node hosts on the same district at approximately 3,000 gears each. Use caution when manually migrating: for example, starting with three nodes in a district then removing one manually can result in the remaining two nodes being unbalanced with 4,000 and 2,000 gears each.

The following steps describe the algorithm for selecting a node on which to place a new gear for an application.

**Algorithm for finding an available node:**

1. Find all the districts.
2. Find the nodes that have the least ***active_capacity***.
3. If enough nodes are found, exclude nodes on which the application already has gears.
4. If districts are enabled and there are nodes that are in districts, exclude nodes which are not in a district.
5. Randomly pick one of the nodes with the lower levels of ***active_capacity***.

Report a bug

### 4.5.3. Enabling Districts

To use districts, the broker MCollective plugin must be configured to enable districts as it is responsible for communicating with node hosts regarding gears.

**Procedure 4.6. Enabling Districts**

- Edit **/etc/openshift/plugins.d/openshift-origin-msg-broker-mcollective.conf** and set the following parameters:

```
DISTRICTS_ENABLED=true
NODE_PROFILE_ENABLED=true
```

Report a bug

### 4.5.4. Creating and Populating Districts

District administration is performed using the **oo-admin-ctl-district** command on the broker host.

**Procedure 4.7. Creating Districts**

1. Create a district using the following command:

```
# oo-admin-ctl-district -c create -n district_name -p gear_profile
```

2. Add a node to the district using the following command:

```
# oo-admin-ctl-district -c add-node -n district_name -i node_hostname
```

The following example shows the creation of a district named **small_district**, to which the node host **`node1.example.com`** is added:

```
# oo-admin-ctl-district -c create -n small_district -p small
Successfully created district: 7521a7801686477f8409e74f67b693f4

{"active_server_identities_size"=>0,
"node_profile"=>"small",
"creation_time"=>"2012-10-24T02:14:48-04:00",
"name"=>"small_district",
"externally_reserved_uids_size"=>0,
"uuid"=>"7521a7801686477f8409e74f67b693f4",
"max_capacity"=>6000,
"available_uids"=>"<6000 uids hidden>",
"max_uid"=>6999,
"available_capacity"=>6000,
"server_identities"=>{}}

# oo-admin-ctl-district -c add-node -n small_district -i node1.example.com
Success!
{"active_server_identities_size"=>1,
"available_capacity"=>6000,
"externally_reserved_uids_size"=>0,
"creation_time"=>"2012-10-24T02:14:48-04:00",
"server_identities"=>{"node1.example.com"=>{"active"=>true}},
"max_capacity"=>6000,
"uuid"=>"7521a7801686477f8409e74f67b693f4",
"available_uids"=>"<6000 uids hidden>",
"node_profile"=>"small",
"max_uid"=>6999,
"name"=>"small_district"}
```

Report a bug

### 4.5.5. Viewing District Information

You can view information about a single district or all available districts on your system. District information is output in JSON format.

**Procedure 4.8. Viewing All Available Districts**

- To view all available districts run the following command:

  ```
  # oo-admin-ctl-district

  { ...
  "uuid"=>"7521a7801686477f8409e74f67b693f4",
  ...}
  ```

**Procedure 4.9. Viewing Single District**

- To view a single district run the following command:

  ```
  # oo-admin-ctl-district -n district_name
  ```

Report a bug

### 4.5.6. Managing District Capacity

Add and remove district capacity to account for changes in district use.

**Procedure 4.10. Adding District Capacity**

Note that there is a hard limit of 6000 gears per district.

- To add 100 gears of capacity run the following command:

```
# oo-admin-ctl-district -c add-capacity -n district_name -s 100
```

**Procedure 4.11. Removing District Capacity**

- To remove 100 gears of capacity run the following command:

```
# oo-admin-ctl-district -c remove-capacity -n district_name -s 100
```

Report a bug

### 4.5.7. Moving Gears Between Nodes

The **oo-admin-move** command moves a gear from one node to another. Note that moving gears requires a **rsync_id_rsa** private key in the broker host's **/etc/openshift/** directory and a matching public key in each node host's **/root/.ssh/authorized_keys** file as explained in the *OpenShift Enterprise Deployment Guide*.

A gear retains its UID when moved, therefore cross-district moves are only allowed when the destination district has the same gear UID available.

**Procedure 4.12. Moving Gears from One Node to Another**

- To move a gear from one node to another, run the following command on the broker host:

```
# oo-admin-move --gear_uuid 3baf79139b0b449d90303464dfa8dd6f -i
node2.example.com

URL: http://app3-demo.example.com
Login: demo
App UUID: 3baf79139b0b449d90303464dfa8dd6f
Gear UUID: 3baf79139b0b449d90303464dfa8dd6f
DEBUG: Source district uuid: NONE
DEBUG: Destination district uuid: NONE
[...]
DEBUG: Starting cartridge 'ruby-1.8' in 'app3' after move on node2.example.com
DEBUG: Fixing DNS and mongo for gear 'app3' after move
DEBUG: Changing server identity of 'app3' from 'node1.example.com' to
'node2.example.com'
DEBUG: The gear's node profile changed from medium to small
DEBUG: Deconfiguring old app 'app3' on node1.example.com after move
Successfully moved 'app3' with gear uuid '3baf79139b0b449d90303464dfa8dd6f'
from 'node1.example.com' to 'node2.example.com'
```

Report a bug

### 4.5.8. Removing Nodes from Districts

If many gears on a node host become idle over time, you may want to compact the district by decommissioning or re-purposing the node host. You can do this by removing the gears from the node host then removing the host from its district by using a combination of the **oo-admin-ctl-district** and **oo-admin-move** commands.

**Procedure 4.13. Removing Nodes from Districts**

The following steps demonstrate an example situation where district **`small_district`** has two node hosts, **`node1.example.com`** and **`node2.example.com`**. The second node host, **`node2.example.com`**, has a high number of idle gears.

1. Run the following commands and fix any problems that are found. This prevents future problems caused by moving a broken gear. On the broker host, run:

   ```
   # oo-admin-chk
   ```

   On the node hosts, run:

   ```
   # oo-accept-node
   ```

2. Deactivate the node you want to remove to prevent applications from being created on or moved to the node. Existing gears continue running. On the broker host, run:

   ```
   # oo-admin-ctl-district -c deactivate-node -n small_district -i
   node2.example.com
   ```

3. Move all the gears from **`node2.example.com`** to **`node1.example.com`** by repeating the following command on the broker host for each gear on **`node2.example.com`**:

   ```
   # oo-admin-move --gear_uuid UUID -i node1.example.com
   ```

4. Remove **`node2.example.com`** from the district:

   ```
   # oo-admin-ctl-district -c remove-node -n small_district -i
   node2.example.com
   ```

Report a bug

### 4.5.9. Removing Districts

When deleting a district, you first remove all the nodes from the district then delete the district.

**Procedure 4.14. Deleting Districts**

1. Set the district's capacity to 0. On the broker host, run:

   ```
   # oo-admin-ctl-district -c remove-capacity -n district_name -s 6000
   ```

2. Remove all the nodes from the district you want to delete by running the following commands for each node:

   ```
   # oo-admin-ctl-district -c deactivate-node -i node_hostname
   # oo-admin-ctl-district -c remove-node -n district_name -i node_hostname
   ```

3. Delete the empty district by running the following command:

   ```
   # oo-admin-ctl-district -c destroy -n district_name
   ```

Report a bug

# Chapter 5. Monitoring

## 5.1. System Monitoring

Report a bug

### 5.1.1. General System Checks

1. Use standard system administration checks to monitor the basic health of your system. For example:
   - ensure adequate memory
   - minimize disk swapping
   - ensure adequate disk space
   - monitor file system health
2. Monitor the services used by OpenShift Enterprise. Ensure the following are running and configured correctly:
   - MCollective
   - Mongo
   - Apache
   - ActiveMQ
   - SELinux and cgroups
3. Use custom scripts to run checks specific to your system. Confirm that the entire system is working by checking:
   - nodes and gears are valid and consistent system-wide by running **`oo-admin-chk`** on a broker host
   - gears are created and deleted correctly
   - available statistics and capacities
   - hosts respond to MCollective using **`mco ping`**

Report a bug

### 5.1.2. Response Times for Administrative Actions

The following sections provide guideline response times for various administrative tasks.

Report a bug

#### 5.1.2.1. Application Creation

Application creation depends on the application type and how long DNS propagation takes. Apart from time spent propagating DNS, creation rarely takes longer than 35 seconds.

Report a bug

#### 5.1.2.2. Restarting Node

The length of time required to perform a node host restart depends on the number of gears on the node and the number of those gears that are active. Node restarts can take between 5-30 minutes.

Report a bug

### 5.1.3. Testing a Path Through the Whole System

A simple test that touches every major component in the system is creating and deleting an application. In an automated monitoring context this requires setting up a test user just for this purpose. These operations test user authentication, the broker application, the MongoDB datastore, the DNS, MCollective and messaging services, and node host and gear functionality.

Report a bug

## 5.2. Broker Monitoring

Monitoring broker activity provides insight into the usage of your OpenShift Enterprise installation and can help diagnose problems with it. Note that problems on the broker only affect actions that interact with the broker, such as application creation. Deployed applications continue to function normally on their nodes even if the broker is unavailable, meaning that developers can still access and update their applications using ssh and git, and applications are still available online.

Report a bug

### 5.2.1. Logs

The following table provides the locations of some important log files and summarizes the information they contain:

**Table 5.1. Broker Log Files**

| File | Description |
|---|---|
| /var/log/openshift/broker/production.log | Logs requests that are processed by the broker application. |
| /var/log/openshift/broker/user_action.log | Logs user actions, including the creation and deletion of gears. Similar to production.log but less verbose. |
| /var/log/openshift/broker/httpd/access_log | Logs calls made to the REST API. |
| /var/log/openshift/broker/httpd/error_log | Logs Rails errors on that occur on start-up. |
| /var/log/openshift/broker/usage.log | Logs gear/filesystem resource usage information if tracking is enabled in **`/etc/openshift/broker.conf`**. |

Report a bug

### 5.2.2. oo-accept-broker

Use the **`oo-accept-broker`** command to check the basic functionality of the broker before testing more intensively.

Report a bug

### 5.2.3. oo-accept-systems

Use the **`oo-accept-systems`** command on a broker host to check that the node hosts have valid settings for use by the broker.

Report a bug

### 5.2.4. oo-admin-chk

Use the **`oo-admin-chk`** command on a broker host to check that all nodes and gears are consistent system-wide.

Report a bug

### 5.2.5. When to Add Additional Brokers

You are unlikely to require more broker hosts for reasons of capacity. The broker is used mainly to create and delete applications, which are sporadic operations. By far the majority of developer and end-user traffic, including updates and actual usage of the applications, is directed to node hosts. For example, the service at https://openshift.redhat.com operates with just four broker hosts in EC2, and there are four mainly for high availability during upgrades. You rarely need more than three or four broker hosts, and only that many to provide high availability.

For information on how to add a broker host, see the *OpenShift Enterprise Deployment Guide*.

Report a bug

## 5.3. Node Monitoring

Problems with node hosts can affect a single gear, several gears, or all the gears on the host. A common indication of a node host problem is an inability to create or remove gears on that host.

Report a bug

### 5.3.1. MCollective Log

MCollective messages are logged in **`/var/log/mcollective.log`** on node hosts. Check this file to confirm proper gear creation.

Report a bug

### 5.3.2. oo-accept-node

Use the **`oo-accept-node`** command to check the basic functionality of the node and correct any failures before testing more intensively.

Report a bug

### 5.3.3. Validating Gears

Validate the gears registered in the MongoDB datastore against those on all node hosts by running **`oo-admin-chk`** on a broker host. This command lists gears that partially failed creation or deletion as well as nodes with incorrect gear counts.

Report a bug

### 5.3.4. Node Capacity

A node host can reach capacity in the amount of storage and RAM available, or in the number of active gears it is hosting. You can add storage and RAM to extend the capabilities of a node, or you can move gears to other nodes that have more resources available.

To determine the active gear capacity of a node host, view the ***`active_capacity`*** entry in **`/etc/mcollective/facts.yaml`**.

To determine the maximum number of active applications on a node host, view the ***`max_active_gears`*** entry in **`/etc/openshift/resource_limits.conf`**.

On a node with 100 applications, 100 active gears out of 100 capacity indicates a full node, while 50/100 is half full. If an application is restored from idle on a full node, the node becomes over-full (101/100). The node continues functioning normally in this situation, but you can move gears to another node to reduce the active applications on the over-full node.

Report a bug

## 5.4. Usage Tracking

You can monitor resource usage per user, including created gears, additional gear storage, and gear age. This feature is enabled in OpenShift Enterprise by default.

As user resource tracking consumes space in the **MongoDB** datastore, it is recommended that you disable tracking if you do not require this feature.

**Procedure 5.1. Disabling Usage Tracking**

1. Open **`/etc/openshift/broker.conf`** on the broker host.
2. Set the value of **`ENABLE_USAGE_TRACKING_DATASTORE`** to **`"false"`**.
   a. To disable audit logging for usage tracking as well, set **`ENABLE_USAGE_TRACKING_AUDIT_LOG`** to **`"false"`**.
3. Restart the broker service.

```
# service openshift-broker restart
```

Report a bug

### 5.4.1. Tracked and Untracked Storage

You can set the amount of tracked and untracked gear storage available to a user with the **`oo-admin-ctl-user`** command. Both types of storage provide additional storage to a user's gears, but untracked storage is not included in usage reports. The total storage available to a user's gear is the sum of the tracked and untracked storage.

When a user adds storage to a gear, their untracked allowance is applied first. When the untracked storage is depleted, further storage is drawn from their tracked allowance.

If you are not tracking resource usage for users, it is recommended to set the maximum untracked storage amount only.

**Procedure 5.2. To set the maximum amount of tracked storage per gear**

- Run the following command on the broker host, replacing ***demo*** with the relevant user name and ***`10`*** with the desired GB of tracked storage.

```
# oo-admin-ctl-user -l demo --setmaxtrackedstorage 10

Setting max_tracked_addtl_storage_per_gear to 10... Done.

User demo:
                  consumed gears: 2
                       max gears: 100
    max tracked storage per gear: 10
  max untracked storage per gear: 0
            plan upgrade enabled:
                      gear sizes: small
            sub accounts allowed: false
```

**Procedure 5.3. To set the maximum amount of untracked storage per gear**

- Run the following command on the broker host, replacing ***demo*** with the relevant user name and ***10*** with the desired GB of untracked storage.

```
# oo-admin-ctl-user -l demo --setmaxuntrackedstorage 10

Setting max_tracked_addtl_storage_per_gear to 10... Done.

User demo:
                  consumed gears: 2
                       max gears: 100
    max tracked storage per gear: 10
  max untracked storage per gear: 10
            plan upgrade enabled:
                      gear sizes: small
            sub accounts allowed: false
```

Report a bug

### 5.4.2. Viewing Accumulated Usage Data

You can view past and current resource usage per user using the **`oo-admin-usage`** command.

**Procedure 5.4. To view resource usage per user**

- Run the following command on the broker host, replacing ***demo*** with the relevant user name.

```
# oo-admin-usage -l demo

Usage for demo (Plan: )
------------------------------------------
#1
 Usage Type: GEAR_USAGE (small)
    Gear ID: 519262ef6892df43f7000001 (racecar)
   Duration: 3 hours and 19 minutes (2013-05-14 12:14:45 - PRESENT)

#2
 Usage Type: ADDTL_FS_GB (3)
    Gear ID: 5192624e6892dfcb3f00000e (foo)
   Duration: 15 seconds (2013-05-14 12:16:33 - 2013-05-14 12:16:48)

#3
 Usage Type: ADDTL_FS_GB (2)
    Gear ID: 5192624e6892dfcb3f00000e (foo)
   Duration: 3 hours and 17 minutes (2013-05-14 12:16:48 - PRESENT)
```

You can also view usage data for all users with the **`oo-admin-ctl-usage`** command.

**Procedure 5.5. To view resource usage for all users**

- Run the following command on the broker host.

```
# oo-admin-ctl-usage --list

Errors/Warnings will be logged to terminal
2013-05-14 15:48:54 -0400 INFO::
---------- STARTED ----------

User: bob
 Gear: 518bcaa26892dfcb74000001, UsageType: GEAR_USAGE, Usage:
23.32543548687111
 Gear: 518bcb876892dfcb74000017, UsageType: GEAR_USAGE, Usage:
23.32543548687111
 Gear: 519254d36892df8f9000000b, UsageType: ADDTL_FS_GB, Usage:
0.05429166666666666
 Gear: 519254d36892df8f9000000b, UsageType: GEAR_USAGE, Usage:
0.08019000000000001
 Gear: 519258d46892df156600001f, UsageType: GEAR_USAGE, Usage:
4.287655764648889
User: demo
 Gear: 5192624e6892dfcb3f00000e, UsageType: ADDTL_FS_GB, Usage: 0.0042325
 Gear: 5192624e6892dfcb3f00000e, UsageType: ADDTL_FS_GB, Usage:
3.5350574313155554
 Gear: 519262ef6892df43f7000001, UsageType: GEAR_USAGE, Usage:
3.5691388202044445
2013-05-14 15:48:54 -0400 INFO::
---------- ENDED, #Errors: 0, #Warnings: 0 ----------
```

Report a bug

# Chapter 6. Command Reference

Report a bug

## 6.1. Broker Administration Commands

These tools are installed with the **`openshift-origin-broker`** and **`openshift-origin-broker-util`** RPMs.

Report a bug

### 6.1.1. oo-accept-broker

This command checks that your broker setup is valid and functional. It is run without options on a broker host.

If there are no errors, it displays **`PASS`** and exits with return code 0. With the **`-v`** option added, it displays the checks that it is performing.

If there are errors, they are displayed, and the return code is the number of errors.

```
# oo-accept-broker -v

INFO: SERVICES: DATA: mongo, Auth: mongo, Name bind
INFO: AUTH_MODULE: rubygem-openshift-origin-auth-mongo
INFO: NAME_MODULE: rubygem-openshift-origin-dns-bind
INFO: Broker package is: openshift-origin-broker
INFO: checking packages
INFO: checking package ruby
INFO: checking package rubygems
INFO: checking package rubygem-rails
INFO: checking package rubygem-passenger
INFO: checking package rubygem-openshift-origin-common
INFO: checking package rubygem-openshift-origin-controller
INFO: checking package openshift-origin-broker
INFO: checking ruby requirements
INFO: checking ruby requirements for openshift-origin-controller
INFO: checking ruby requirements for config/application
INFO: checking firewall settings
INFO: checking services
INFO: checking datastore
INFO: checking cloud user authentication
INFO: auth plugin = /var/www/openshift/broker/config/initializers/broker.rb:2:
uninitialized constant ApplicationObserver (NameError) from -:6
INFO: checking dynamic dns plugin
INFO: checking messaging configuration
PASS
```

Report a bug

### 6.1.2. oo-accept-systems

This command checks that node host **`PUBLIC_HOSTNAME`** and **`PUBLIC_IP`** configuration settings are valid and unique system-wide. It also checks the cartridges installed on the nodes and the status of the broker's cache. It is run without options on the broker host.

If there are no errors, the command displays **`PASS`** and exits with return code 0. With the **`-v`** option added, it displays the checks that it is performing.

If there are errors, they are displayed, and the return code is the number of errors.

```
# oo-accept-systems -v

INFO: checking that each public_hostname resolves to external IP
INFO: PUBLIC_HOSTNAME node1.example.com for node2.example.com resolves to
10.4.59.136
INFO: PUBLIC_HOSTNAME node2.example.com for node1.example.com resolves to
10.4.59.133
INFO: checking that each public_hostname is unique
INFO: checking that public_ip has been set for all nodes
INFO: PUBLIC_IP 10.4.59.136 for node1.example.com
INFO: PUBLIC_IP 10.4.59.133 for node2.example.com
INFO: checking that public_ip is unique for all nodes
INFO: checking that all node hosts have cartridges installed
INFO: cartridges for node1.example.com: cron-1.4|ruby-1.9|perl-5.10|jenkins-client-
1.4|diy-0.1|jenkins-1.4|php-5.3|haproxy-1.4|abstract|abstract-jboss|jbosseap-
6.0|mysql-5.1|postgresql-8.4|ruby-1.8|jbossews-1.0|python-2.6|abstract-httpd
INFO: cartridges for node2.example.com: diy-0.1|jenkins-client-1.4|cron-
1.4|jbossews-1.0|php-5.3|abstract-httpd|ruby-1.9|python-2.6|jbosseap-6.0|perl-
5.10|abstract|postgresql-8.4|abstract-jboss|ruby-1.8|jenkins-1.4|haproxy-1.4|mysql-
5.1
INFO: checking that same cartridges are installed on all node hosts
INFO: checking that broker's cache is not stale
INFO: API reports carts: diy-0.1|jbossews-1.0|php-5.3|ruby-1.9|python-2.6|jbosseap-
6.0|perl-5.10|ruby-1.8|jenkins-1.4|jenkins-client-1.4|cron-1.4|postgresql-
8.4|haproxy-1.4|mysql-5.1
PASS
```

Report a bug

### 6.1.3. oo-admin-chk

This command checks that application records in the MongoDB datastore are consistent with gear presence on the node hosts. With the **-v** option added, it displays the checks that it is performing.

```
# oo-admin-chk -v

Started at: 2013-05-03 03:36:28 +1000
Time to fetch mongo data: 0.005s
Total gears found in mongo: 3
Time to get all gears from nodes: 20.298s
Total gears found on the nodes: 3
Total nodes that responded : 1
Checking application gears and ssh keys on corresponding nodes:
51816f026892dfec74000004 : String... OK
518174556892dfec74000040 : String... OK
518176826892dfec74000059 : String... OK
Checking node gears in application database:
51816f026892dfec74000004... OK
518174556892dfec74000040... OK
518176826892dfec74000059... OK
Success
Total time: 20.303s
Finished at: 2013-05-03 03:36:49 +1000
```

With the **-l** option added, additional levels of checks can be included, as described in the following **--help** output:

```
# oo-admin-chk --help

== Synopsis

/usr/sbin/oo-admin-chk: Check all user applications

== Usage

/usr/sbin/oo-admin-chk OPTIONS

Options:
-v|--verbose
    Print information about each check being performed
-l|--level [0, 1, 2]
    Level '0' is default, and checks to see if any gears are present in mongo but
don't exist on the nodes and vice-versa.
    Level '1' performs extra checks such as integrity of consumed_gears count
    Level '2' additionally performs checks for application data integrity in mongo
and checks for unused and unreserved gear UIDs
-h|--help
    Show Usage info
```

Report a bug

### 6.1.4. oo-admin-repair

This command checks for and fixes various inconsistencies in the MongoDB datastore on the broker. For example, as a mismatch in SSH keys can be a potential security risk, the tool fixes any found by taking what is in the broker datastore and pushing it to the gear. Refer to the following **`--help`** output for additional uses:

```
# oo-admin-repair --help

== Synopsis

/usr/sbin/oo-admin-repair:  Utility to check and fix various inconsistencies in
mongo data

The following issues can be fixed with this script:
  - mismatch between user's consumed gears and actual gears across all
domains/applications for the user
  - mismatch between the ssh keys in mongo and the ssh keys on the gear on the node
  - mismatch in available UIDs for a district and actual UIDs used by the gears on
the nodes within the district

== Usage

/usr/sbin/oo-admin-repair OPTIONS

Options:
-v|--verbose
    Print information about each check being performed
-r|--report-only
    Only report the mismatches, don't fix them
--consumed-gears
    Fix  mismatch in user's consumed gears vs actual gears in mongo
--ssh-keys
    Fix  mismatch in SSH keys between mongo and on the node for a gear
--district-uids
    Fix mismatch in available UIDs for a district in mongo
-h|--help
    Show Usage info
```

Report a bug

### 6.1.5. oo-register-dns

This command updates DNS A records in BIND by wrapping an **`nsupdate`** command. Normally this command is used for broker or node hosts, although it can be used for other infrastructure hosts. Do not use this command to change DNS records for applications and gears, as these are CNAME records.

```
# oo-register-dns --help

== Synopsis

oo-register-dns: Register node's DNS name with Bind
  This command must be run as root.

== Usage

oo-register-dns --with-node-hostname node1 \
               --with-node-ip 192.168.0.1 \
               --domain example.com

== List of arguments
 -h|--with-node-hostname    host        Hostname for the node (required)
 -n|--with-node-ip          ip          IP of the node (required)
 -d|--domain                domain      Domain name for this node (optional,
default: example.com)
 -s|--dns-server            server      IP address or hostname of DNS server to
update (optional, default: 127.0.0.1)
 -k|--key-file              file        Bind key (optional, default:
/var/named/<domain name>.key)
 -?|--help                              Print this message
```

Report a bug

### 6.1.6. oo-admin-move

This command moves a gear from one node to another. Note that moving gears requires the **`rsync_id_rsa`** private key in the broker host's **`/etc/openshift/`** directory and the public key in each node host's **`/root/.ssh/authorized_keys`** file as explained in the *OpenShift Enterprise Deployment Guide*.

A gear retains its Unix UID when moved, therefore cross-district moves are only allowed when the destination district has the same gear UID available.

```
# oo-admin-move --help

== Synopsis

oo-admin-move: Move an app from one node to another

== Usage

oo-admin-move OPTIONS

Options:
--gear_uuid <gear_uuid>
    Gear uuid to move
--destination_district_uuid <district_uuid>
   Destination district uuid
-i|--target_server_identity <server_identity>
   Target server identity
-p|--node_profile <node_profile>
   Node profile
-t|--timeout
   timeout
--change_district
   Move to a different district other than the source district
-h|--help
   Show Usage info
```

**Procedure 6.1. Moving Gears from One Node to Another**

- To move a gear with UUID ***3baf79139b0b449d90303464dfa8dd6f*** from node host **node1.example.com** to **node2.example.com**, run the following command on the broker host:

  ```
  # oo-admin-move --gear_uuid 3baf79139b0b449d90303464dfa8dd6f -i
  node2.example.com

  URL: http://app3-demo.example.com
  Login: demo
  App UUID: 3baf79139b0b449d90303464dfa8dd6f
  Gear UUID: 3baf79139b0b449d90303464dfa8dd6f
  DEBUG: Source district uuid: NONE
  DEBUG: Destination district uuid: NONE
  [...]
  DEBUG: Starting cartridge 'ruby-1.8' in 'app3' after move on node2.example.com
  DEBUG: Fixing DNS and mongo for gear 'app3' after move
  DEBUG: Changing server identity of 'app3' from 'node1.example.com' to
  'node2.example.com'
  DEBUG: The gear's node profile changed from medium to small
  DEBUG: Deconfiguring old app 'app3' on node1.example.com after move
  Successfully moved 'app3' with gear uuid '3baf79139b0b449d90303464dfa8dd6f'
  from 'node1.example.com' to 'node2.example.com'
  ```

Report a bug

### 6.1.7. oo-admin-broker-auth

This command recreates broker authentication keys. If **AUTH_SALT** is changed in **/etc/openshift/broker.conf**, restart the broker service and run the **oo-admin-broker-auth** command to recreate authentication tokens for all applicable gears.

```
# oo-admin-broker-auth --help

== Synopsis

oo-admin-broker-auth: Recreate the authentication tokens for gears.

To rekey all tokens in parallel use --rekey-all or pipe the UUIDs in on STDIN.  An
individual Gear can be rekeyed with the --rekey flag.

== Usage

oo-admin-broker-auth OPTIONS

Options:
--rekey-all
    Rekey all Gears using Broker authentication tokens
--rekey [uuid]
-t|--timeout [integer]
    How long to wait for MCollective Node discovery
-v|--verbose
    Show debugging information
-q|--quiet
    Show as little output as possible
--find-gears
    Print out gears with using Broker key authentication
--dryrun
    Show what would be done.  Useful for finding discrepancies in Nodes reported
by the Broker datastore and the message bus
-h|--help
    Show Usage info
```

Report a bug

### 6.1.8. oo-admin-broker-cache

This command clears the broker cache, the Management Console cache, or both. See Section 3.1, “Clearing Broker and Console Application Cache” for more information.

```
# oo-admin-broker-cache --help

== Synopsis

oo-admin-broker-cache: Manage the broker Rails application's cache

== Usage

oo-admin-broker-cache OPTIONS

Options:
-c|--clear
    Remove all entries from the broker Rails application's cache
--console
    Also clear the cache for the developer console if installed
-q|--quiet
    Show as little output as possible
-h|--help
    Show Usage info
```

Report a bug

### 6.1.9. oo-admin-ctl-district

This command performs district operations. See Section 4.5, “Managing Districts” for more information on districts.

```
# oo-admin-ctl-district --help

== Synopsis

oo-admin-ctl-district: Control districts

== Usage

oo-admin-ctl-district OPTIONS

Options:
-u|--uuid     <district uuid>
    District uuid  (alphanumeric)
-c|--command <command>
    (add-node|remove-node|deactivate-node|activate-node|add-capacity|remove-
capacity|create|destroy)
-n|--name <district name>
    District name (Used on create or in place of uuid on other commands)
-p|--node_profile <gear_size>
    (small(default)) Only needed for create
-i|--server_identity
    Node server_identity (required)
-s|--size
    Size to add or remove (positive number) (required)
-b|--bypass
    Ignore warnings
-h|--help
    Show usage info
```

**Procedure 6.2. Adding Nodes**

- To add a node run the following command:

```
# oo-admin-ctl-district -c add-node -n district_name -i node_hostname
```

**Procedure 6.3. Deleting Nodes**

- To delete a node run the following commands:

```
# oo-admin-ctl-district -c deactivate-node -i node_hostname
# oo-admin-ctl-district -c remove-node -n district_name -i node_hostname
```

**Procedure 6.4. Deactivating Nodes**

- To deactivate a node run the following command:

```
# oo-admin-ctl-district -c deactivate-node -i node_hostname
```

**Procedure 6.5. Reactivating Nodes**

- To reactivate a node run the following command:

```
# oo-admin-ctl-district -c activate-node -n district_name -i node_hostname
```

Report a bug

### 6.1.10. oo-admin-ctl-user

This command administers users on the system. Some features are disabled for user accounts by default, such as the ability to add additional storage to gears or add private SSL certificates to aliases, and require this tool in order to enable them or set an explicit allowed value for the user.

See Section 5.4.1, “Tracked and Untracked Storage” for more information on setting maximum tracked and untracked storage per gear.

See Section 4.2, “Setting Gear Quotas for Specific Users” for more information on setting gear quotas.

```
# oo-admin-ctl-user --help
== Synopsis

oo-admin-ctl-user: Control user settings.

== Notes

  *** WARNING *** WARNING *** WARNING *** WARNING ***

  DO NOT USE THIS SCRIPT TO MODIFY A LOT OF USERS AT ONCE!

  *** WARNING *** WARNING *** WARNING *** WARNING ***

== Usage

oo-admin-ctl-user OPTIONS

Options:
  -l|--login <login_name>
    Login with OpenShift access (required)
  --setmaxgears <number>
    Set the maximum number of gears a user is allowed to use
  --setmaxtrackedstorage <number>
    Set the maximum additional storage per gear that will be tracked for a user
  --setmaxuntrackedstorage <number>
    Set the maximum additional storage per gear that will be untracked for a user
  --setconsumedgears <number>
    Set the number of gears a user has consumed (use carefully)
  --listsubaccounts
    List the subaccounts that have been created under this parent account (login)
  --addsubaccount <subaccount login>
    The sub account to add to the login parent account
  --removesubaccount <subaccount login>
    The sub account to remove from the login parent account
  --allowsubaccounts (true|false)
    Add / Remove the capability to manage sub accounts
  --allowplanupgrade (true|false)
    Add / Remove the capability to upgrade your plan.
  --allowprivatesslcertificates (true|false)
    Add / Remove the capability to add private SSL certificates
  --addgearsize <gearsize>
    Add gearsize to the capability for this login user
  --removegearsize <gearsize>
    Remove gearsize from the capability for this login user
  --inheritgearsizes (true|false)
    Allow / Disallow inheritance of login user gearsizes capability to sub accounts
  -h|--help
    Show Usage info

Examples:
  List the current user settings with:
    oo-admin-ctl-user -l bob@redhat.com

  Set the maximum number of gears a user is allowed to use with:
    oo-admin-ctl-user -l bob@redhat.com --setmaxgears 10
```

Report a bug

### 6.1.11. oo-admin-ctl-authorization

This command enables you to delete either all users' expired authorization tokens or all users' valid and expired authorization tokens.

**Procedure 6.6. Deleting All Users' Expired Authorization Tokens**

- Run the following command on the broker host:

```
# oo-admin-ctl-authorization -c expire

Deleted expired authorizations: 3
```

**Procedure 6.7. Deleting All Users' Valid and Expired Authorization Tokens**

- Run the following command on the broker host:

```
# oo-admin-ctl-authorization -c revoke_all

Deleted all authorizations: 6
```

Report a bug

#### 6.1.12. oo-admin-ctl-domain

This command is used to query and control a user's domain. It produces detailed output in YAML format.

```
# oo-admin-ctl-domain --help

== Synopsis

oo-admin-ctl-domain: Manage user domains

== Usage

oo-admin-ctl-domain OPTIONS

Options:
-l|--login <login_name>
    Login with OpenShift access (required)
-n|--namespace <Namespace>
    Namespace for application(s) (alphanumeric - max 16 chars) (required)
-c|--command (create|update|delete|info|env_add|env_del)
-s|--ssh_key <ssh key>
    User's SSH key
-t|--key_type <ssh key type>
    User's SSH key type (ssh-rsa|ssh-dss|ecdsa-sha2-nistp256-cert-
v01@openssh.com|ecdsa-sha2-nistp384-cert-v01@openssh.com|ecdsa-sha2-nistp521-cert-
v01@openssh.com|ssh-rsa-cert-v01@openssh.com|ssh-dss-cert-v01@openssh.com|ssh-rsa-
cert-v00@openssh.com|ssh-dss-cert-v00@openssh.com|ecdsa-sha2-nistp256|ecdsa-sha2-
nistp384|ecdsa-sha2-nistp521)
-k|--key_name <ssh key name>
    User's SSH key name
-e|--env_name <env var name>
-v|--env_value <env var value>
-h|--help:
    Show Usage info
```

Report a bug

### 6.1.13. oo-admin-ctl-app

This command provides administration command options for applications.

```
# oo-admin-ctl-app --help

== Synopsis

oo-admin-ctl-app: Control user applications

== Usage

oo-admin-ctl-app OPTIONS

Options:
-l|--login <login_name>
    Login with OpenShift access (required)
-a|--app     <application>
    Application name  (alphanumeric) (required)
-c|--command <command>
    (start|stop|force-stop|restart|status|destroy|force-destroy|remove-gear|remove-
cartridge) (required)
-b|--bypass
    Ignore warnings
--gear_uuid
    Gear uuid to operate on
--cartridge
    Cartridge to operate on
-h|--help
    Show Usage info
```

Report a bug

## 6.2. Node Administration Commands

These tools are installed on node hosts with the **`openshift-origin-node-util`** RPM.

**Note**

Node hosts do not have administrative access to other nodes or to brokers, so running the commands described in this section only affect the node on which they are run.

Report a bug

### 6.2.1. oo-accept-node

This command checks that your node setup is valid and functional and that its gears are in good condition. It is run without options on a node host.

If there are no errors, it displays **`PASS`** and exits with return code 0. With the **`-v`** option added, it displays the checks that it is performing.

If there are errors, they are displayed, and the return code is the number of errors.

```
# oo-accept-node -v

INFO: loading node configuration file /etc/openshift/node.conf
INFO: loading resource limit file /etc/openshift/resource_limits.conf
INFO: checking selinux status
INFO: checking selinux origin policy
INFO: checking selinux booleans
INFO: checking package list
INFO: checking services
INFO: checking kernel semaphores >= 512
INFO: checking cgroups configuration
INFO: checking presence of /cgroup
INFO: checking presence of /cgroup/all
INFO: checking presence of /cgroup/all/openshift
INFO: checking filesystem quotas
INFO: checking quota db file selinux label
INFO: checking 54 user accounts
INFO: checking application dirs
INFO: checking system httpd configs
PASS
```

Report a bug

### 6.2.2. oo-idler-stats

This command displays basic statistics about gears on a node.

```
# oo-idler-stats --help

Usage: oo-idler-stats [options]

Options:
 -h, --help     show this help message and exit
 -v, --verbose  Print additional details.
 --validate     Perform additional sanity checks.
```

Report a bug

### 6.2.3. oo-admin-ctl-gears

This command is used to control gears on a node host. It is used by the openshift-gears service at boot time to activate existing gears and can also be used manually by an administrator.

```
# oo-admin-ctl-gears

Usage: /usr/sbin/oo-admin-ctl-gears
{startall|stopall|status|restartall|condrestartall|idleall|unidleall|startgear
[uuid]|stopgear [uuid]|restartgear [uuid]|idlegear [gear]|unidlegear
[gear]|list|listidle}

  list: simply all gears on the node.
  status: shows status of all gears on the node.
  startall: starts all gears, one by one.
  stopall: stops all gears, one by one.
  restartall: restarts all gears, one by one (NOT the same as stopall/startall)
  condrestartall: like restartall, but uses a lockfile to keep from being run
concurrently with another instance of itself.
  startgear X: starts individual gear X
  stopgear X: stops individual gear X
```

Report a bug

### 6.2.4. Idler Commands

The idler is a tool for shutting down gears that have not been used recently in order to reclaim their resources and overcommit the node host's resource usage.

Report a bug

#### 6.2.4.1. oo-idler

This command stops an application, forwards the application's URL to **`/var/www/html/restorer.php`**, and records the application's status as **`idled`**.

```
Usage: /usr/sbin/oo-idler
  -u UUID       idles the app
  -l            list all idled apps
```

Report a bug

#### 6.2.4.2. oo-restorer

**`restorer.php`** runs this command to restart an application's gear when the application is accessed. **`oo-restorer`** can also be run manually. Normally a web request is in the wrong context to restart a gear and httpd service, so oddjob is used to send a request to **`oo-restorer`** from the correct context. Restoring requires the oddjobd and messagebus services to be running.

```
# oo-restorer

Usage: /usr/bin/oo-restorer
 -u UUID  (app to restore UUID)
```

Report a bug

#### 6.2.4.3. oo-last-access

This command records in the gear operations directory the time since the last web or git access to an application. Running this command regularly in a cron job allows automatic idling of stale gears.

An example auto-idler cron script:

```
# run the last-access compiler hourly
0 * * * * /usr/bin/oo-last-access > /var/lib/openshift/last_access.log 2>&1
# run the auto-idler twice daily and idle anything stale for 5 days
30 7,19 * * * /usr/bin/oo-autoidler 5
```

Report a bug

#### 6.2.4.4. oo-list-stale

This command retrieves a list of gears that are older than a given age. The default age is 15 days.

Report a bug

#### 6.2.4.5. oo-autoidler

This command uses **`oo-list-stale`** to retrieve a list of stale gears, then runs **`oo-idler`** to idle them. It is recommended that this command be run regularly in a cron job.

Report a bug

# Revision History

| | | |
|---|---|---|
| **Revision 1.2-1** | **Tue Aug 6 2013** | **Brian Moss** |

BZ985670: Incorrect command for deleting a user's namespace.

| | | |
|---|---|---|
| **Revision 1.2-0** | **Tue Jul 9 2013** | **Brian Moss** |

OpenShift Enterprise 1.2 Release.
BZ921583: DEFAULT_GEAR_CAPABILITIES available in OSE broker conf.
BZ905699: UUID should be uid in find available node algorithm description.
BZ907741: Administrator cannot create a district with a custom gear size until it is included in VALID_GEAR_SIZE.
Document oo-accept-systems script.
Update oo-admin-move content.
Update oo-admin-ctl-user content.
Update oo-admin-chk content.
Update Clearing Broker Application Cache and Command Reference sections with oo-admin-broker-cache usage.
Add oo-admin-broker-auth to Command Reference section.
Update Broker Log Files table with new locations.
Add Usage Monitoring section.
Update oo-admin-ctl-user section for latest changes.
Add oo-admin-ctl-authorization to Command Reference section.
Add section on custom/community carts.
Document oo-admin-repair script.
BZ949643: OSE 1.2 Administration Guide needs to tell how to clear the console's cartridge cache
Add Setting Gear Supplementary Groups section.
Update gear placement algorithm.