



Red Hat Fuse 7.0

Managing Fuse

Managing Fuse applications with Hawtio and Prometheus

Red Hat Fuse 7.0 Managing Fuse

Managing Fuse applications with Hawtio and Prometheus

Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

When you deploy a Fuse application, you can use both the Fuse Console (Hawtio) and Prometheus to monitor and interact with Red Hat Fuse integrations.

Table of Contents

| | |
|---|-----------|
| PREFACE | 4 |
| PART I. ACCESSING THE FUSE CONSOLE | 5 |
| CHAPTER 1. ACCESSING THE FUSE CONSOLE ON OPENSIFT | 6 |
| 1.1. OPENSIFT TEMPLATES FOR FUSE CONSOLE | 6 |
| 1.2. DEPLOYING THE FUSE CONSOLE ON YOUR OPENSIFT CLUSTER | 6 |
| 1.2.1. Using the OpenShift Console | 6 |
| 1.2.2. Using the command line | 7 |
| CHAPTER 2. ACCESSING THE FUSE CONSOLE FOR SPRING BOOT STANDALONE | 8 |
| CHAPTER 3. ACCESSING THE FUSE CONSOLE FOR JBOSS EAP | 9 |
| 3.1. RESOLVING DELAYED FUSE CONSOLE LOADING | 9 |
| CHAPTER 4. ACCESSING THE FUSE CONSOLE FOR KARAF | 10 |
| 4.1. SECURING FUSE CONTAINERS ON KARAF | 10 |
| 4.1.1. Enabling SSL/TLS security | 10 |
| 4.1.2. Controlling user access to the Fuse Console | 10 |
| CHAPTER 5. DISABLING THE FUSE CONSOLE | 13 |
| 5.1. KARAF | 13 |
| 5.2. JBOSS EAP | 13 |
| 5.3. SPRING BOOT | 13 |
| 5.4. OPENSIFT | 13 |
| PART II. VIEWING AND MANAGING FUSE APPLICATIONS | 14 |
| CHAPTER 6. VIEWING CONTAINERS AND APPLICATIONS (FUSE ON OPENSIFT) | 15 |
| CHAPTER 7. CONNECTING TO REMOTE FUSE INTEGRATIONS (STANDALONE DISTRIBUTIONS) | 16 |
| 7.1. UNLOCKING THE FUSE CONSOLE | 16 |
| 7.2. ALLOWING CONNECTIONS TO REMOTE FUSE INSTANCES | 17 |
| 7.3. CONNECTING TO A REMOTE JOLOKIA INSTANCE | 17 |
| 7.4. SETTING DATA MOVING PREFERENCES | 18 |
| 7.5. VIEWING JVM RUNTIME INFORMATION | 19 |
| CHAPTER 8. VIEWING AND MANAGING APACHE CAMEL APPLICATIONS | 20 |
| 8.1. OVERVIEW | 20 |
| 8.2. INTERACTING WITH A CAMEL APPLICATION | 20 |
| CHAPTER 9. VIEWING AND MANAGING JMX DOMAINS AND MBEANS | 24 |
| CHAPTER 10. VIEWING AND MANAGING YOUR OSGI ENVIRONMENT (KARAF STANDALONE) | 25 |
| 10.1. VIEWING AND MONITORING THREAD STATE | 25 |
| CHAPTER 11. VIEWING LOG ENTRIES | 26 |
| 11.1. SETTING FUSE CONSOLE LOG ATTRIBUTES | 26 |
| CHAPTER 12. CHANGING THE FUSE CONSOLE BRANDING | 27 |
| CHAPTER 13. ENSURING THAT DATA DISPLAYS CORRECTLY IN THE FUSE CONSOLE | 28 |
| PART III. ACCESSING PROMETHEUS | 29 |
| 1. OPENSIFT SETUP | 29 |
| 2. INSTALLING PROMETHEUS | 29 |

| | |
|--|----|
| 3. CONFIGURING PROMETHEUS | 30 |
| 4. CONTROLLING THE METRICS THAT PROMETHEUS MONITORS AND COLLECTS | 30 |
| 5. VIEWING PROMETHEUS METRICS WITH GRAFANA | 31 |
| 6. GENERATING ALERTS | 31 |

PREFACE

Red Hat Fuse provides two enterprise monitoring tools for viewing and managing Fuse integrations:

- The Fuse Console is a web-based console that you access from a browser to monitor and manage a running Fuse container. The Fuse Console is based on Hawtio open source software (<http://hawtio.io/>).
- Prometheus (<https://prometheus.io/docs/introduction/overview/>) stores system and integration-level metrics for Fuse distributions. You can use a graphical analytics interface, such as Grafana, to view and analyze the stored historical data.

The audience for this guide is Red Hat Fuse administrators. This guide assumes that you are familiar with the Red Hat Fuse platform, Apache Camel, and the processing requirements for your organization.

When you deploy a Fuse application, you can use both the Fuse Console (Hawtio) and Prometheus to monitor and interact with Red Hat Fuse integrations.

Fuse Console

The Fuse Console provides a central interface to examine and manage the details of one or more deployed Fuse containers. You can also monitor Red Hat Fuse and system resources, perform updates, and start or stop services.

The Fuse Console is available when you install Red Hat Fuse standalone or use Fuse on OpenShift. The integrations that you can view and manage in the Fuse Console depend on the plugins that are running. Possible plugins include:

- Camel
- JMX
- Spring Boot
- OSGI
- Runtime
- Logs

The Fuse Console is a web-based console. For a list of supported browsers, go to <https://access.redhat.com/articles/310603>.

Prometheus

Prometheus is container-native software built for storing historical data and for monitoring large, scalable systems of which Fuse on OpenShift is a component. It gathers data over an extended time, rather than just for the currently running session.

PART I. ACCESSING THE FUSE CONSOLE

How you access the Fuse Console depends on your Fuse distribution:

- [Fuse on OpenShift](#)
- Fuse Standalone:
 - [Spring Boot](#)
 - [Red Hat JBoss EAP](#)
 - [Apache Karaf](#)

CHAPTER 1. ACCESSING THE FUSE CONSOLE ON OPENSIFT



NOTE

Security and user management for the Fuse Console is handled by OpenShift. Role-based access control is not yet available for Fuse on OpenShift.

For a Fuse on OpenShift distribution, deploy the Fuse Console using the corresponding OpenShift template.

1.1. OPENSIFT TEMPLATES FOR FUSE CONSOLE

Table 1.1, “Fuse Console templates” describes the two OpenShift templates that you can use to access the Fuse Console, depending on the type of Fuse application deployment.

Table 1.1. Fuse Console templates

| Type | Template | Description |
|-----------|--|--|
| cluster | fis-console-cluster-template.json | Use an OAuth client that requires the cluster-admin role to be created. The Fuse Console can discover and connect to Fuse applications deployed across multiple namespaces or projects. |
| namespace | fis-console-namespace-template.json | Use a service account as OAuth client, which only requires the admin role in a project to be created. This restricts the Fuse Console access to this single project, and as such acts as a single tenant deployment. |

Optionally, you can view a list of the template parameters by running the following command:

```
oc process --parameters -f fis-console-namespace-template.json
```

1.2. DEPLOYING THE FUSE CONSOLE ON YOUR OPENSIFT CLUSTER

You can deploy the Fuse Console from the OpenShift Console or from the command line.

1.2.1. Using the OpenShift Console

To deploy the Fuse Console on your OpenShift cluster, follow these steps (where **myproject** is the name of your project):

1. If the Fuse Console templates are not already installed on OpenShift, run the following command to create the template as a resource in OpenShift:

```
oc create -n openshift -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-000099-redhat-5/fis-console-namespace-template.json
```

- In the OpenShift console, select **Add to Project** → **Select from Project** to see the Fuse Console template.

1.2.2. Using the command line

To deploy the Fuse Console from the command line:

- Create a new application based on a Fuse Console template by running one of the following commands (where **myproject** is the name of your project):

- For the Fuse Console **cluster** template, where **myhost** is the hostname:

```
oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-000099-redhat-5/fis-console-cluster-template.json -p ROUTE_HOSTNAME=myhost
```

- For the Fuse Console **namespace** template:

```
oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-000099-redhat-5/fis-console-namespace-template.json
```



NOTE

You can omit the ROUTE_HOSTNAME parameter for the *namespace* template because OpenShift automatically generates one.

- Obtain the status of your deployment by running this command:

```
oc status
```

Here is an example response:

```
In project myproject on server https://192.168.64.12:8443
https://fuse-console.192.168.64.12.nip.io (redirects) (svc/fuse70-console-service)
dc/fuse70-console deploys openshift/jboss-fuse70-console:1.0
deployment #1 deployed 2 minutes ago - 1 pod
```

- To access the Fuse Console from a browser, use the provided URL (for example, <https://fuse-console.192.168.64.12.nip.io>).

CHAPTER 2. ACCESSING THE FUSE CONSOLE FOR SPRING BOOT STANDALONE

To access the Fuse Console for a standalone Fuse Spring Boot distribution:

1. Add **hawtio-springboot** to your Fuse application's **pom.xml** file dependencies.

```
<dependency>  
  <groupId>io.hawt</groupId>  
  <artifactId>hawtio-springboot</artifactId>  
</dependency>
```

Note that you do not need to specify the version because it is provided by the Maven BOM.

2. Edit the **src/main/resources/application.properties** file and set **endpoints.jolokia.sensitive** to false:

```
endpoints.jolokia.sensitive = false
```

3. Run the Fuse application by using the following command:

```
mvn spring-boot:run
```

4. To determine the port number for the Fuse Console URL, obtain the **management.port** value by looking at the value set in the **src/main/resources/application.properties** file. For example:

```
management.port = 10001
```

5. To open the Fuse Console in a browser, use the following URL syntax where **nnnnn** is the value of the **management.port** property:

<http://localhost:nnnnn/hawtio/index.html>

For example: <http://localhost:10001/hawtio/index.html>

CHAPTER 3. ACCESSING THE FUSE CONSOLE FOR JBOSS EAP

Before you can access the Fuse Console for Red Hat JBoss Enterprise Application Platform, you must install Fuse on the JBoss EAP container. For step-by-step instructions, see [Installing on JBoss EAP](#).

To access the Fuse Console for a standalone JBoss EAP distribution:

1. Start Red Hat Fuse standalone with the following command:
On Linux/Mac OS: **./bin/standalone.sh**

On Windows: **./bin/standalone.bat**
2. In a web browser, enter the URL to connect to the Fuse Console. For example:
<http://localhost:8080/hawtio>
3. In the login page, enter your user name and password and then click **Log In**.

By default, the Fuse Console shows the Home page. The left navigation tabs indicate the running plugins.

3.1. RESOLVING DELAYED FUSE CONSOLE LOADING

If the main Fuse Console page takes a long time to display in the browser, you might need to reduce the number and the size of the log files. You can use the **periodic-size-rotating-file-handler** to rotate the file when it reaches a maximum size (rotate-size) and maintains a number of files (max-backup-index). For details on how to use this handler, see the Red Hat JBoss Enterprise Application Platform product documentation.

CHAPTER 4. ACCESSING THE FUSE CONSOLE FOR KARAF

To access the Fuse Console for Apache Karaf standalone:

1. Install Fuse on the Karaf container. For step-by-step instructions, see [Installing on Apache Karaf](#).
2. In the command line, navigate to the directory in which you installed Red Hat Fuse and run the following command to start Fuse standalone:

```
./bin/fuse
```

The Karaf console starts and shows version information, the default Fuse Console URL, and a list of common commands.

3. In a browser, enter the URL to connect to the Fuse Console. For example: <http://localhost:8181/hawtio>
4. In the login page, enter your user name and password and then click **Log In**.

By default, the Fuse Console shows the Home page. The left navigation tabs indicate the running plugins.

4.1. SECURING FUSE CONTAINERS ON KARAF

You can implement the following security features to secure Fuse containers on Apache Karaf:

- Enable SSL/TLS security
- Configure user authorization

4.1.1. Enabling SSL/TLS security

SSL/TLS security is not enabled by default for the Fuse Console. It is recommended that you enable SSL/TLS security on the Fuse Console to protect username/password credentials from snooping. For detailed instructions on how to enable SSL/TLS security, see the [Security Guide](#).

4.1.2. Controlling user access to the Fuse Console



NOTE

For this release, role-based access control for the Fuse Console is only enabled for Fuse on Karaf standalone.

The operations that an authenticated user are allowed to perform depend on the role (or roles) assigned to that user, as listed in [Table 4.1, "Role-based access on Karaf standalone"](#).

Ensure that you have the necessary user role authorization to perform the Fuse Console operations that you want to perform.

To set a user role:

1. Open the Red Hat Fuse **etc/users.properties** file in an editor.

2. Add an entry for the user name, password, and role.
For example, the following entry in the **etc/users.properties** file defines the admin user and grants the admin role.

```
admin = secretpass,admin
```

3. Save the file.

Table 4.1. Role-based access on Karaf standalone

| Operation | admin | manager | viewer |
|---|-------|---------|--------|
| login/logout | Y | Y | Y |
| View Help topics | Y | Y | Y |
| Set user preferences | Y | Y | Y |
| Connect | | | |
| Discover and connect to remote integrations | Y | Y | Y |
| Discover and connect to local integrations | Y | Y | Y |
| Camel | | | |
| View all running Camel applications | Y | Y | Y |
| Start, suspend, resume, and delete Camel Contexts | Y | Y | N |
| Send messages | Y | Y | N |
| Add endpoints | Y | Y | N |
| View routes, route diagrams, and runtime statistics | Y | Y | Y |
| Start and stop routes | Y | Y | N |
| Delete routes | Y | Y | N |
| JMX | | | |

| Operation | admin | manager | viewer |
|--|-------|---------|--------|
| Change attribute values | Y | Y | N |
| Select and view attributes in a time-based chart | Y | Y | Y |
| View operations | Y | Y | Y |
| OSGI | | | |
| View bundles, features, packages, services, servers, framework, and configurations | Y | Y | Y |
| Add and delete bundles | Y | Y | N |
| Add configurations | Y | Y | N |
| Install and uninstall features | Y | Y | N |
| Runtime | | | |
| View system properties, metrics, and threads | Y | Y | Y |
| Logs | | | |
| Viewing logs | Y | Y | Y |

For more information on role-based access control, see [Deploying into Apache Karaf](#).

CHAPTER 5. DISABLING THE FUSE CONSOLE

How you disable the Fuse Console depends on your Fuse distribution.

5.1. KARAF

To disable the Fuse Console on Karaf so that it becomes inaccessible to all users without affecting any other component, follow these steps:

1. To determine the hawtio-web bundle ID, use the following command to list the Fuse bundles that the Fuse Console uses:
osgi:list | grep hawtio
2. To stop the bundle, use the **osgi:stop** command. For example, if the **hawtio :: Web console** bundle has an ID of 246, type this command:
osgi:stop 246

The bundle goes into the resolved state and you can no longer access the Fuse Console.

For more information about managing bundles, see the "Lifecycle Management" chapter of [Deploying into Apache Karaf](#).

5.2. JBOSS EAP

To disable the Fuse Console on JBoss EAP, do one of the following:

- Remove the deployment file: **\$EAP_HOME/standalone/deployments/hawtio-wildfly-xxxxx.war**
- Undeploy the Fuse Console by using the JBoss EAP admin console or command line interface.

5.3. SPRING BOOT

For Spring Boot, the Fuse Console is not enabled by default.

5.4. OPENSIFT

For OpenShift, the Fuse Console is not enabled by default.

PART II. VIEWING AND MANAGING FUSE APPLICATIONS

You can view and manage Fuse applications as described in the following sections:

- [Chapter 6, *Viewing containers and applications \(Fuse on OpenShift\)*](#)
- [Chapter 7, *Connecting to remote Fuse integrations \(standalone distributions\)*](#)
- [Chapter 8, *Viewing and managing Apache Camel applications*](#)
- [Chapter 9, *Viewing and managing JMX domains and MBeans*](#)
- [Chapter 10, *Viewing and managing your OSGI environment \(Karaf standalone\)*](#)
- [Chapter 11, *Viewing log entries*](#)
- [Chapter 12, *Changing the Fuse Console branding*](#)
- [Chapter 13, *Ensuring that data displays correctly in the Fuse Console*](#)

CHAPTER 6. VIEWING CONTAINERS AND APPLICATIONS (FUSE ON OPENSIFT)

When you login to the Fuse Console for OpenShift, the Fuse Console home page shows the available containers.

To manage (create, edit, or delete) containers, use the OpenShift console.

To view Fuse applications on the OpenShift cluster, click the **Online** tab.

CHAPTER 7. CONNECTING TO REMOTE FUSE INTEGRATIONS (STANDALONE DISTRIBUTIONS)

The Fuse Console uses Jolokia, an agent-based approach to Java Management Extensions (JMX) that requires extra software (an agent) installed on the client. By default, Red Hat Fuse includes a jolokia agent.

With standalone Fuse Console distributions, you can connect to remote integrations that already have a jolokia agent (<https://jolokia.org/>) running inside them. If the process that you want to connect to does not have a jolokia agent inside, refer to the jolokia documentation (<http://jolokia.org/agent.html>).

7.1. UNLOCKING THE FUSE CONSOLE

By default, Jolokia for Fuse 7 standalone (on Apache Karaf and JBoss EAP) is locked and the Fuse Console is not accessible remotely.

To unlock the Fuse Console for a hostname or IP address other than **localhost** or **127.0.0.1**, follow these steps:

1. Open the `jolokia-access.xml` file in an editor.
On Karaf, the XML file is located in the `$KARAF_HOME/etc` folder.

On JBoss EAP, it is located in the `$EAP_HOME/standalone/configuration` folder.
2. Register the hostnames or IP addresses for the Fuse integrations that you want to access with the Fuse console by adding them to the `<cors>` section.
For example, to access hostname `0.0.0.3` from the Fuse Console, add the `<allow-origin>http://0.0.0.3:*</allow-origin>` line as shown:

```
<!--
Cross-Origin Resource Sharing (CORS) restrictions

By default, only CORS access within localhost is allowed for maximum security.

You can add trusted hostnames in the <cors> section to unlock CORS access from them.

-->

<cors>

  <!-- Allow cross origin access only within localhost -->

  <allow-origin>http*://localhost:*</allow-origin>

  <allow-origin>http*://127.0.0.1:*</allow-origin>

  <allow-origin>http://0.0.0.3:*</allow-origin>

  <!-- Whitelist the hostname patterns as <allow-origin> -->

  <!--

  <allow-origin>http*://*.example.com</allow-origin>
```

```

<allow-origin>http*://*.example.com:*</allow-origin>

-->

<!-- Check for the proper origin on the server side to protect against CSRF -->

<strict-checking />

</cors>

```

3. Save the file.

7.2. ALLOWING CONNECTIONS TO REMOTE FUSE INSTANCES

The Fuse Console's proxy servlet uses whitelist host protection, with which by default the Fuse Console can only connect to localhost. If you want to connect the Fuse Console to other remote Fuse instances, you need to configure the whitelist as follows:

- For Apache Karaf, make the following configuration changes in **etc/system.properties** file:

```
hawtio.proxyWhitelist = localhost, 127.0.0.1, myhost1, myhost2, myhost3
```

- For JBoss EAP, make the following configuration changes in the **standalone/configuration/standalone-*.xml** file:

```
<property name=hawtio.proxyWhitelist" value="localhost, 127.0.0.1, myhost1, myhost2,
myhost3"/>
```

- For Spring Boot, configure the **hawtio.proxyWhitelist** system property in the **main()** method of your Spring Boot application:

```
System.setProperty("hawtio.proxyWhitelist", "localhost, 127.0.0.1, myhost1, myhost2,
myhost3");
```

7.3. CONNECTING TO A REMOTE JOLOKIA INSTANCE

Before you begin, you need to know the host name, port, and path of the jolokia agent to be able to connect.

To connect to a remote Jolokia instance so that you can examine its JVM:

1. Click the **Connect** tab.
2. Click the **Remote** tab, and then **Add connection**.

Add Connection
✕

The fields marked with * are required.

* Name

* Scheme

* Host

Port

Path

3. Type the **Name**, **Scheme** (HTTP or HTTPS), and the **hostname**.
4. Click **Test Connection**.
5. Click **Add**.



NOTE

The Fuse Console automatically probes the local network interfaces other than localhost and 127.0.0.1 and adds them to the whitelist. Hence, you do not need to manually register the local machine's addresses to the whitelist.

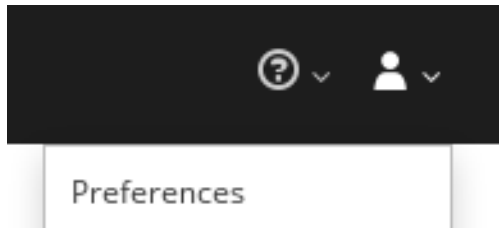
7.4. SETTING DATA MOVING PREFERENCES

You can change the following Jolokia preferences, for example, if you want to more frequently refresh data that displays in the Fuse Console. Note that increasing the frequency of data updates impacts networking traffic and increases the number of requests made to the server.

- **Update rate** - The period between polls to Jolokia to fetch JMX data (the default is 5 seconds).
- **Maximum depth** - The number of levels that Jolokia will marshal an object to JSON on the server side before returning (the default is 7).
- **Maximum collection size** - The maximum number of elements in an array that Jolokia marshals in a response (the default is 50,000).

To change the values of these settings:

1. In the upper right of the Fuse Console, click the user icon and then click **Preferences**.



2. Edit the options and then click **Close**.

7.5. VIEWING JVM RUNTIME INFORMATION

To view JVM runtime information, such as system properties, metrics, and threads, click the **Runtime** tab.

CHAPTER 8. VIEWING AND MANAGING APACHE CAMEL APPLICATIONS

In the Fuse Console's **Camel** tab, you view and manage Apache Camel contexts, routes, and dependencies.

8.1. OVERVIEW

The **Camel** tab is only available when you connect to a container that uses one or more Camel routes.

You can view the following details:

- A list of all running Camel contexts
- Detailed information of each Camel context such as Camel version number and runtime statics
- Lists of all routes in each Camel application and their runtime statistics
- Graphical representation of the running routes along with real time metrics

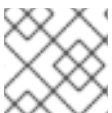
You can also interact with a Camel application by:

- Starting and suspending contexts
- Managing the lifecycle of all Camel applications and their routes, so you can restart, stop, pause, resume, etc.
- Live tracing and debugging of running routes
- Browsing and sending messages to Camel endpoints

8.2. INTERACTING WITH A CAMEL APPLICATION

To start, suspend, or delete a context:

1. In the **Camel** tab's tree view, click **Camel Contexts**.
2. Check the box next to one or more contexts in the list.
3. Click **Start** or **Suspend**.
4. To delete a context, you must first stop it. Then click the ellipse icon and select **Delete** from the dropdown menu.



NOTE

When you delete a context, you remove it from the deployed application.

To view Camel application details:

1. In the **Camel** tab's tree view, click a Camel application.
2. To view a list of application attributes and values, click **Attributes**.

3. To view a graphical representation of the application attributes::
 - a. Click **Chart**.
 - b. Click **Edit** to select the attributes that you want to see in the chart.
4. To view inflight and blocked exchanges, click **Exchanges**.
5. To view application endpoints, click **Endpoints**. You can filter the list by **URL**, **Route ID**, and **direction**.
6. Click **Type Converters** to view, enable, and disable statistics related to the Camel built-in type conversion mechanism, that is used to convert message bodies and message headers to different types.
7. Click **Operations** to view and execute JMX operations, such as adding or updating routes from XML or finding all Camel components available in the classpath.

To interact with Camel routes:

1. In the **Camel** tab's tree view, click the application's routes folder to view a list of the routes:

Routes

| Start Stop ⋮ | | |
|--------------------------|---------|---------|
| <input type="checkbox"/> | Name ^ | State |
| <input type="checkbox"/> | _route1 | Started |
| <input type="checkbox"/> | _route2 | Started |

2. To start, stop, or delete one or more routes:
 - a. Check the box next to one or more routes in the list.
 - b. Click **Start** or **Stop**.
 - c. To delete a route, you must first stop it. Then click the ellipse icon and select **Delete** from the dropdown menu.

Routes

| Start Stop ⋮ | | |
|-------------------------------------|--------|--------|
| <input checked="" type="checkbox"/> | Name ^ | Delete |
| <input checked="" type="checkbox"/> | | |

**NOTE**

- When you delete a route, you remove it from the deployed application.
- You can also select a specific route in the tree view and then click the upper-right menu to start, stop, or delete it.

3. To view a graphical diagram of the routes, click **Route Diagram**.
4. To view inflight and blocked exchanges, click **Exchanges**.
5. To view endpoints, click **Endpoints**. You can filter the list by URL, Route ID, and direction.
6. Click **Type Converters** to view, enable, and disable statistics related to the Camel built-in type conversion mechanism, which is used to convert message bodies and message headers to different types.

To interact with a specific route:

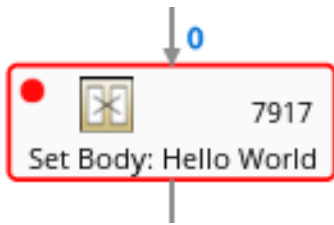
1. In the **Camel** tab's tree view, select a route.
2. To view a list of route attributes and values, click **Attributes**.
3. To view a graphical representation of the route attributes, click **Chart**. You can click **Edit** to select the attributes that you want to see in the chart.
4. To view inflight and blocked exchanges, click **Exchanges**.
5. Click **Operations** to view and execute JMX operations on the route, such as dumping the route as XML or getting the route's Camel ID value.

To trace messages through a route:

1. In the **Camel** tab's tree view, select a route.
2. Select **Trace**, and then click **Start tracing**.
3. To send messages to a route:
 - a. In the **Camel** tab's tree view, open the context's endpoints folder and then select an endpoint.
 - b. Click the **Send** subtab.
 - c. Configure the message in JSON or XML format.
 - d. Click **Send**.
4. Return to the route's **Trace** tab to view the flow of messages through the route.

To debug a route:

1. In the **Camel** tab's tree view, select a route.
2. Select **Debug**, and then click **Start debugging**.
3. To add a breakpoint, select a node in the diagram and then click **Add breakpoint**. A red dot appears in the node:



The node is added to the list of breakpoints:

| Breakpoints | |
|-------------|---|
| setBody1 | × |
| log1 | × |

4. Click the down arrow to step to the next node or the **Play** button to resume running the route.
5. Click the **Pause** button to suspend all threads for the route.
6. Click **Stop debugging** when you are done. All breakpoints are cleared.

CHAPTER 9. VIEWING AND MANAGING JMX DOMAINS AND MBEANS

Java Management Extensions (JMX) is a Java technology that allows you to manage resources (services, devices, and applications) dynamically at runtime. The resources are represented by objects called MBeans (for Managed Bean). You can manage and monitor resources as soon as they are created, implemented, or installed.

With the JMX plugin on the Fuse Console, you can view and manage JMX domains and MBeans. You can view MBean attributes, run commands, and create charts that show statistics for the MBeans.

The **JMX** tab provides a tree view of the active JMX domains and MBeans organized in folders. You can view details and execute commands on the MBeans.

To view and edit MBean attributes:

1. In the tree view, select an MBean.
2. Click the **Attributes** tab.
3. Click an attribute to see its details.

To perform operations:

1. In the tree view, select an MBean.
2. Click the **Operations** tab, expand one of the listed operations.
3. Click **Execute** to run the operation.

To view charts:

1. In the tree view, select an item.
2. Click the **Chart** tab.

CHAPTER 10. VIEWING AND MANAGING YOUR OSGI ENVIRONMENT (KARAF STANDALONE)

For Apache Karaf standalone distributions, you can view and manage the Red Hat Fuse OSGi environment. You can view and manage container bundles, features, and configurations, as well as Java packages and OSGi services.

The **OSGi** tab contains a series of subtabs with options for each container component:

Bundles

List of installed bundles. You can install and uninstall bundles, start and stop bundles, and edit bundle properties. You can also filter the list and toggle between list and grid view.

Features

List of available features. You can install and uninstall features or feature repositories, and drill down to view feature details.

Packages

List of installed Java packages. You can view package versions and associated bundles.

Services

List of running services. You can view service IDs, associated bundles and object classes.

Declarative Services

List of declarative OSGi services. You can view the service state and drill down to view service details. You can also activate and deactivate services.

Server

Detailed information about the local or remote host in read-only mode.

Framework

Configuration options for the container OSGi framework. You can set the framework start level and the initial bundle start level.

Configuration

List of configuration objects. You can view the state of each object and drill down to view or edit object details. You can also create a new configuration object.

10.1. VIEWING AND MONITORING THREAD STATE

To view and monitor the state of threads:

1. Click the **Runtime** tab and then the **Threads** subtab. The **Threads** page lists active threads and stack trace details for each thread. By default, the thread list shows all threads in descending ID order.
2. To sort the list by increasing ID, click the **ID** column label.
3. Optionally, filter the list by thread state (for example, **Blocked**) or by thread name.
4. To drill down to detailed information for a specific thread, such as the lock class name and full stack trace for that thread, in the **Actions** column, click **More**.

CHAPTER 11. VIEWING LOG ENTRIES

You can view log entries for Red Hat Fuse in the **Logs** tab. The **Logs** tab is available (for Fuse standalone on Karaf and Fuse standalone on JBoss EAP) when the Java application has the Log MBean available .

You can filter the list of logs to show specific log types, and drill down to each log entry to view detailed information about the log entry.

The **Logs** tab contains the following sections:

Action Bar

Options to filter the log entries section according to a text string or the logging level.

Log Entries

List view of the log entries. By default, the list shows log entries in ascending order. You can change the default sorting in the **User → Preferences → Server Logs** page. Click the log entry link to drill down to details about the log entry, such as the bundle name, thread, and the full message text.

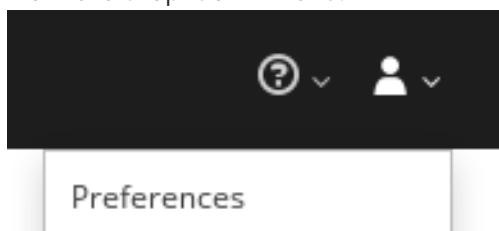
11.1. SETTING FUSE CONSOLE LOG ATTRIBUTES

You can customize the following Fuse Console browser settings for storing log messages:

- The number of log statements to keep in the Fuse Console (the default is 100).
- The global log level: **INFO** (the default), OFF, ERROR, WARN, and DEBUG.
- The child-level Hawtio messages to include, such as hawtio-oauth and hawtio-core-utils.

To change the defaults:

1. In the upper right corner of the Fuse Console, click the user icon and then click **Preferences** from the drop-down menu:



2. Edit the options and then click **Close**.

To reset the Console Logs settings to the default values, click **Reset → Reset settings**.

CHAPTER 12. CHANGING THE FUSE CONSOLE BRANDING

To change the branding name and image for the Fuse Console, you can edit the following files:

- The **hawtconfig.json** file in the Fuse Console's war file (**karaf-install-dir/system/io/hawt/hawtio-war/version/hawtio-war-version.war**):

```
{
  "branding": {
    "appName": "Hawtio Management Console",
    "appLogoUrl": "img/hawtio-logo.svg",
    "companyLogoUrl": "img/hawtio-logo.svg"
  },
  "about": {
    "title": "Hawtio Management Console",
    "productInfo": [],
    "additionalInfo": "",
    "copyright": "",
    "imgSrc": "img/hawtio-logo.svg"
  },
  "disabledRoutes": []
}
```

- The war file's style sheet (**.css**) file to modify other aspects of the Fuse Console UI.



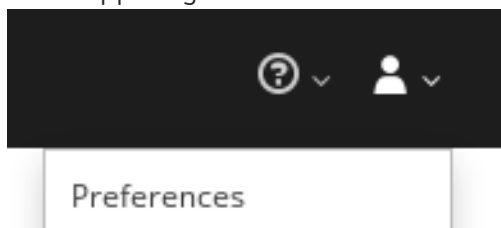
NOTE

If you have already run the Fuse Console in a web browser, the branding is stored in the browser's local storage. To use new branding settings, you must clear the browser's local storage.

CHAPTER 13. ENSURING THAT DATA DISPLAYS CORRECTLY IN THE FUSE CONSOLE

If the display of the queues and connections in the Fuse Console is missing queues, missing connections, or displaying inconsistent icons, adjust the Jolokia collection size parameter that specifies the maximum number of elements in an array that Jolokia marshals in a response.

1. In the upper right corner of the Fuse Console, click the user icon and then click **Preferences**.



2. Increase the value of the **Maximum collection size** option (the default is 50,000).
3. Click **Close**.

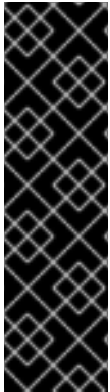
PART III. ACCESSING PROMETHEUS



NOTE

For this release, Prometheus is only supported for Fuse on OpenShift. For documentation about Prometheus on OpenShift Container Platform, go to:

https://access.redhat.com/documentation/en-us/openshift_container_platform/3.9/html/installation_and_configuration/install-config-cluster-metrics#openshift-prometheus



IMPORTANT

Prometheus on OpenShift Container Platform is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information on Red Hat Technology Preview features support scope, see <https://access.redhat.com/support/offerings/techpreview/>.

You can use Prometheus to monitor and store Fuse on OpenShift data by exposing an endpoint with your Fuse application's data to Prometheus format. Prometheus stores the data so that you can use a graphical tool, such as Grafana, to visualize and run queries on the data.

1. OPENSIFT SETUP

To use Prometheus to monitor Fuse on OpenShift you must:

- Install the Ansible Service Broker (ASB), which requires OpenShift v3.9 or later. To install the Ansible Service Broker, follow the instructions here: <https://github.com/openshift/ansible-service-broker>
- When you start OpenShift, use the `--service-catalog` option.

2. INSTALLING PROMETHEUS

The Ansible Playbook Bundle uses an image of Prometheus:

<https://github.com/ansibleplaybookbundle/prometheus-apb>

1. To install the Ansible Playbook Bundle (APB), run these commands:

```
wget https://raw.githubusercontent.com/openshift/ansible-service-broker/master/scripts/run_latest_build.sh
chmod +x run_latest_build.sh
./run_latest_build.sh
```

2. To give the developer account permissions to install, run the following commands:

```
oc create -n ansible-service-broker -f
https://raw.githubusercontent.com/ansibleplaybookbundle/ansible-playbook-bundle/apb-
1.1.6-1/templates/openshift-permissions.template.yaml
```

```
oc export cm/broker-config -n ansible-service-broker | sed 's/sandbox_role: ./sandbox_role:
\admin\" | oc replace -f - cm/broker-config -n ansible-service-broker
```

The Prometheus APB should appear.

3. Follow the provided instructions. Optionally, you can install Grafana.

For information about getting started with Prometheus, go to:

https://prometheus.io/docs/prometheus/latest/getting_started/

3. CONFIGURING PROMETHEUS

You can set the following environment variables in OpenShift in order to configure your application's Prometheus agent:

Table 3. Prometheus Environment Variables

| Environment Variable | Description | Default |
|-------------------------------|--|---|
| AB_PROMETHEUS_HOST | The host address to bind. | 0.0.0.0 |
| AB_PROMETHEUS_OFF | If set, disables the activation of Prometheus (echoes an empty value). | Prometheus is enabled. |
| AB_PROMETHEUS_PORT | The Port to use. | 9779 |
| AB_JMX_EXPORTER_CONFIG | Uses the file (including path) as the Prometheus configuration file. | The /opt/prometheus/prometheus-config.yml file with Camel metrics |
| AB_JMX_EXPORTER_OPTS | Additional options to append to the JMX exporter configuration. | Not applicable. |

4. CONTROLLING THE METRICS THAT PROMETHEUS MONITORS AND COLLECTS

By default, Prometheus uses a configuration file (<https://raw.githubusercontent.com/jboss-fuse/application-templates/master/prometheus/prometheus-config.yml>) that includes all possible metrics exposed by Camel.

If you have custom metrics within your application that you want Prometheus to monitor and collect (for example, the number of orders that your application processes), you can use your own configuration file. Note that the metrics that you can identify are limited to those supplied in JMX.

To use a custom configuration file to expose JMX beans that are not covered by the default configuration, follow these steps:

1. Create a custom Prometheus configuration file. You can use the contents of the default file (**prometheus-config.yml** <https://raw.githubusercontent.com/jboss-fuse/application-templates/master/prometheus/prometheus-config.yml>) as a guide for the format. You can use any name for the custom configuration file, for example: **my-prometheus-config.yml**
2. Add your prometheus configuration file (for example, **my-prometheus-config.yml**) to your application's **src/main/fabric8-includes** directory.
3. Create a **src/main/fabric8/deployment.xml** file within your application and add an entry for the **AB_JMX_EXPORTER_CONFIG** environment variable with its value set to your configuration file. For example:

```
env:
  name: AB_JMX_EXPORTER_CONFIG
  value: "my-prometheus-config.yml"
```

This environment variable applies to your application at the pod level.

4. Rebuild and deploy your application.

5. VIEWING PROMETHEUS METRICS WITH GRAFANA

Grafana is an analytics platform that allows you to query, visualize, alert on and understand your metrics using dashboards.

To view Prometheus metrics with Grafana, you need to:

- Install Grafana: <https://prometheus.io/docs/visualization/grafana/#installing>
- Configure a prometheus data source in Grafana (see "Creating a Prometheus data source" : <https://prometheus.io/docs/visualization/grafana/>)
- Use Prometheus metrics in your own dashboards and graphs (see "Creating a Prometheus graph" : <https://prometheus.io/docs/visualization/grafana/>)

6. GENERATING ALERTS

For an example of using Prometheus for OpenShift to generate alerts, see the Red Hat Cloud Forms *Monitoring, Alerts, and Reporting* guide:

https://access.redhat.com/documentation/en-us/red_hat_cloudforms/4.6/html/monitoring_alerts_and_reporting/integrating_prometheus_alerts