Red Hat

Red Hat Fuse 7.6

Managing Fuse

Managing Fuse applications with the Fuse Console

Last Updated: 2020-08-20

Red Hat Fuse 7.6 Managing Fuse

Managing Fuse applications with the Fuse Console

Abstract

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

Table of Contents

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 (https://hawt.io/). This guide describes how to use the Fuse Console.

- 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. To learn more about using Prometheus, see Monitoring Red Hat Integration .

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.

CHAPTER 1. MONITORING AND MANAGING RED HAT FUSE APPLICATIONS ON OPENSHIFT

1.1. ABOUT THE FUSE CONSOLE

The Red Hat Fuse Console is a web console based on HawtIO open source software. For a list of supported browsers, go to Supported Configurations.

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

1.2. SETTING UP THE FUSE CONSOLE ON OPENSHIFT 4.X

On OpenShift 4.x, setting up the Fuse Console involves securing, installing, and deploying it.

First, you must generate a client certificate so that you can secure the Fuse Console as described in Section 1.2.1, “Generating a certificate to secure the Fuse Console on OpenShift 4.x” .

After you generate the client certificate, you have these options for installing and deploying the Fuse Console:

- Section 1.2.2, “Installing and deploying the Fuse Console on OpenShift 4.x by using the OperatorHub”
  You can use the Fuse Console Operator to install and deploy the Fuse Console so that it has access to Fuse applications in a specific namespace.

- Section 1.2.3, “Installing and deploying the Fuse Console on OpenShift 4.x by using the command line”
  You can use the command line and one of the Fuse Console templates to install and deploy the Fuse Console so that it has access to Fuse applications in multiple namespaces on the OpenShift cluster or in a specific namespace.

NOTE

- User management for the Fuse Console is handled by OpenShift.
- Role-based access control (for users accessing the Fuse Console after it is deployed) is not yet available for Fuse on OpenShift.

1.2.1. Generating a certificate to secure the Fuse Console on OpenShift 4.x

On OpenShift 4.x, to keep the connection between the Fuse Console proxy and the Jolokia agent secure, you must generate a client certificate before you deploy the Fuse Console. You must use the service signing certificate authority private key to sign the client certificate.

IMPORTANT

You must generate and sign a separate client certificate for each OpenShift cluster. Do not use the same certificate for more than one cluster.

Prerequisites

- You have **cluster admin** access to the OpenShift cluster.
- If you are generating certificates for more than one OpenShift cluster and you previously generated a certificate for a different cluster in the current directory, do one of the following to ensure that you generate a different certificate for the current cluster:
  - Delete the existing certificate files (for example, **ca.crt**, **ca.key**, and **ca.srl**) from the current directory.
  - Change to a different working directory. For example, if your current working directory is named **cluster1**, create a new **cluster2** directory and change your working directory to it:
    mkdir ../cluster2

    cd ../cluster2

Procedure

1. Login to OpenShift as a user with cluster admin access:

   ```
   oc login -u <user_with_cluster_admin_role>
   ```

2. Retrieve the service signing certificate authority keys, by executing the following commands:

   - To retrieve the certificate:

     ```
     oc get secrets/signing-key -n openshift-service-ca -o "jsonpath={.data['tls\.crt']}" | base64
     --decode > ca.crt
     ```

   - To retrieve the private key:

     ```
     oc get secrets/signing-key -n openshift-service-ca -o "jsonpath={.data['tls\.key']}" |
     base64 --decode > ca.key
     ```

3. Generate the client certificate, as documented in Kubernetes certificates administration, using either **easyrsa**, **openssl**, or **cfssl**.
   Here are the example commands using openssl:

   a. Generate the private key:

   ```
   openssl genrsa -out server.key 2048
   ```

   b. Write the CSR config file.

   ```
   cat <<EOT >> csr.conf
     [ req ]
     default_bits = 2048
     prompt = no
     default_md = sha256
     distinguished_name = dn

     [ dn ]
     CN = fuse-console.fuse.svc

     [ v3_ext ]
     authorityKeyIdentifier=keyid,issuer:always
     keyUsage=keyEncipherment,dataEncipherment,digitalSignature
     extendedKeyUsage=serverAuth,clientAuth
   EOT
   ```

   c. Generate the CSR:

   ```
   openssl req -new -key server.key -out server.csr -config csr.conf
   ```

   d. Issue the signed certificate:

   ```
   openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt
   -days 10000 -extensions v3_ext -extfile csr.conf
   ```

Next steps

You need this certificate to create the secret for the Fuse Console as described in the following sections, depending on how you want to install the Fuse Console:

- By using the Fuse Console Operator
- By using the command line

1.2.2. Installing and deploying the Fuse Console on OpenShift 4.x by using the OperatorHub

To install the Fuse Console on OpenShift 4.x, you can use the Fuse Console Operator provided in the OpenShift OperatorHub. To deploy the Fuse Console, you create an instance of the installed operator.

Prerequisite

- You have **cluster admin** access to the OpenShift cluster.

- You have generated a client certificate for the Fuse Console as described in Generating a certificate to secure the Fuse Console on OpenShift 4.x.

Procedure

To install and deploy the Fuse Console:

1. Log in to the OpenShift console in your web browser as a user with **cluster admin** access.
2. Select **Home** > **Projects**, and then select the project to which you want to add the Fuse Console.
3. Click **Operators** and then click **OperatorHub**.
4. In the search field window, type **Fuse Console** to filter the list of operators.
5. Click **Fuse Console Operator**.
6. In the Fuse Console Operator install window, click **Install**.
   The **Create Operator Subscription** form opens.
   - For **Installation mode**, you install the Fuse Console Operator to a specific namespace (the current OpenShift project).
     Note that after you install the operator, you can then choose to deploy the Fuse Console to monitor applications in all namespaces on the cluster or to monitor applications only in the namespace in which the Fuse Console operator is installed.
   - For the **Approval Strategy**, you can select **Automatic** or **Manual** to configure how OpenShift handles updates to the Fuse Console Operator.
     - If you select **Automatic** updates, when a new version of the Fuse Console Operator is available, the OpenShift Operator Lifecycle Manager (OLM) automatically upgrades the running instance of the Fuse Console without human intervention.
     - If you select **Manual** updates, when a newer version of an Operator is available, the OLM creates an update request. As a cluster administrator, you must then manually approve that update request to have the Fuse Console Operator updated to the new version.
7. Click **Subscribe**.
   OpenShift installs the Fuse Console Operator in the current namespace.
8. To verify the installation, click **Operators** and then click **Installed Operators**. You can see the Fuse Console in the list of operators.
9. In a Terminal window, use the certificate that you generated in Securing the Fuse Console on OpenShift 4.x to create the secret and mount it in the Fuse Console by using the following command where **APP_NAME** is the name of the Fuse Console **Deployment** (for example, **fuse-console**).

   ```
   oc create secret tls APP_NAME-tls-proxying --cert server.crt --key server.key
   ```

   For example, if the name of the Fuse Console application is **fuse-console**, type this command:

   ```
   oc create secret tls fuse-console-tls-proxying --cert server.crt --key server.key
   ```

   If successful, this command returns a response that confirms the secret was created, for example:

```
secret/fuse-console-operator-tls-proxying created
```

10. To deploy the Fuse Console by using the OpenShift web console:

    a. In the list of **Installed Operators**, under the **Name** column, click **Fuse Console**.

    b. On the **Overview** page under **Provided APIs**, click **Create Instance**. A new Custom Resource Definition (CRD) file opens.
    By default, the Fuse Console is deployed to the current namespace.

    c. If you want to deploy the Fuse Console to discover and manage applications within the current namespace, skip to the next step.
    Optionally, if you want to deploy the Fuse Console to discover and manage applications across all namespaces that are on the cluster (and for which you are an authenticated user), edit the CRD file by changing the value of the **spec.type** field from **namespace** to **cluster**.

    d. Click **Create**.
    The **Fuse Console CRD** page opens showing the new Fuse Console service.

11. To open the Fuse Console:

    a. Obtain the Fuse Console URL:

       - From within the OpenShift web console, select **Networking** > **Routes**.
       - From the command line, type the **oc get routes** command.

    b. In your web browser, open the Fuse Console URL and then log in.
    An **Authorize Access** page opens in the browser listing the required permissions.

    c. Click **Allow selected permissions**.
    The Fuse Console opens in the browser and shows the Fuse application pods that you have authorization to access.

12. Click **Connect** for the application that you want to view.
    A new browser window opens showing the application in the Fuse Console.

1.2.3. Installing and deploying the Fuse Console on OpenShift 4.x by using the command line

On OpenShift 4.x, you can choose one of these deployment options to install and deploy the Fuse Console from the command line:

- **cluster** - The Fuse Console can discover and connect to Fuse applications deployed across multiple namespaces (projects) on the OpenShift cluster. To deploy this template, you must have the administrator role for the OpenShift cluster.
- **namespace** - The Fuse Console has access to a specific OpenShift project (namespace). To deploy this template, you must have the administrator role for the OpenShift project.

To view a list of the parameters for the Fuse Console templates, run the following OpenShift command:

```
oc process --parameters -f https://raw.githubusercontent.com/jboss-fuse/application-
templates/application-templates-2.1.0.fuse-760043-redhat-00003/fuse-console-namespace-os4.json
```

Prerequisites

- Before you install and deploy the Fuse Console, you must generate a client certificate that is signed with the service signing certificate authority as described in Generating a certificate to secure the Fuse Console on OpenShift 4.x.
- You have the **cluster admin** role for the OpenShift cluster.
- The Fuse Console image stream (along with the other Fuse image streams) are installed, as described in Installing Fuse imagestreams and templates on the OpenShift 4.x server .

Procedure

1. Verify that the Fuse Console image stream is installed by using the following command to retrieve a list of all templates:

   ```
   oc get template -n openshift
   ```

2. Optionally, if you want to update the already installed image stream with new release tags, use the following command to import the Fuse Console image to the **openshift** namespace:

   ```
   oc import-image fuse7/fuse7-console:1.6 --from=registry.redhat.io/fuse7/fuse-console:1.6 --
   confirm -n openshift
   ```

3. Obtain the Fuse Console **APP_NAME** value by running the following command:

   - For the cluster template:

     ```
     oc process --parameters -f https://raw.githubusercontent.com/jboss-fuse/application-
     templates/application-templates-2.1.0.fuse-760043-redhat-00003/fuse-console-cluster-
     os4.json
     ```

   - For the namespace template:

     ```
     oc process --parameters -f https://raw.githubusercontent.com/jboss-fuse/application-
     templates/application-templates-2.1.0.fuse-760043-redhat-00003/fuse-console-
     namespace-os4.json
     ```

4. From the certificate that you generated in Securing the Fuse Console on OpenShift 4.x, create the secret and mount it in the Fuse Console by using the following command (where **APP_NAME** is the name of the Fuse Console application).

   ```
   oc create secret tls APP_NAME-tls-proxying --cert server.crt --key server.key
   ```

5. Create a new application by running the following command (where **$project** is the name of your OpenShift project, **$APP_NAME** is name of the application obtained in the above step, and **$DOMAIN_NAME** is the hostname to access the Fuse Console:

   - For the cluster template:

     ```
     oc new-app -n $project -f {templates-base-url}/fuse-console-cluster-os4.json -p
     ROUTE_HOSTNAME=$APP_NAME.$DOMAIN_NAME -p APP_NAME=$APP_NAME
     ```

   - For the namespace template:

```
oc new-app -n myproject -f {templates-base-url}/fuse-console-namespace-os4.json
```

6. Obtain the status and the URL of your Fuse Console deployment by running this command:

   ```
   oc status
   ```

7. To access the Fuse Console from a browser, use the URL that is returned in Step 6 (for example, https://fuse-console.192.168.64.12.nip.io).

1.2.4. Upgrading the Fuse Console on OpenShift 4.x

Red Hat OpenShift 4.x handles updates to operators, including the Red Hat Fuse operators. For more information see the Operators OpenShift documentation.

In turn, operator updates can trigger application upgrades, depending on how the application is configured.

For Fuse Console applications, you can also trigger an upgrade to an application by editing the **.spec.version** field of the application custom resource definition.

Prerequisite

- You have OpenShift cluster admin permissions.

Procedure

To upgrade a Fuse Console application:

1. In a terminal window, use the following command to change the **.spec.version** field of the application custom resource definition:

   ```
   oc patch <project-name> <custom-resource-name> --type='merge' -p '{"spec":
   {"version":"1.7.1"}}'
   ```

   For example:

   ```
   oc patch myproject example-fuseconsole --type='merge' -p '{"spec":{"version":"1.7.1"}}'
   ```

2. Check that the application’s status has updated:

   ```
    oc get myproject
   ```

   The response shows information about the application, including the version number:

   ```
   NAME                AGE   URL                                        IMAGE
   example-fuseconsole   1m    https://fuseconsole.192.168.64.38.nip.io
   docker.io/fuseconsole/online:1.7.1
   ```

   When you change the value of the **.spec.version** field, OpenShift automatically redeploys the application.

3. To check the status of the redeployment that is triggered by the version change:

   ```
   oc rollout status deployment.v1.apps/example-fuseconsole
   ```

▪

A successful deployment shows this response:

```
deployment "example-fuseconsole" successfully rolled out
```

1.3. SETTING UP THE FUSE CONSOLE ON OPENSHIFT 3.11

On OpenShift 3.11, you can set up the Fuse Console in two ways:

- By adding the *centralized* Fuse Console catalog item to a project so that you can monitor all the running Fuse containers in the project.
- From a specific pod so that you can monitor that single running Fuse container.

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

NOTE

To install Fuse Console on Minishift or CDK based enviroments, follow the steps explained in the KCS article below.

- To install Fuse Console on Minishift or CDK based enviroments, see KCS 4998441.
- If it is necessary to disable Jolokia authentication see the workaround described in KCS 3988671.

Prerequisites

- Install the Fuse on OpenShift image streams and the templates for the Fuse Console as described in Fuse on OpenShift Guide.
- For cluster mode on OpenShift 3.11, you need the cluster admin role and the cluster mode template. Run the following command:

```
oc adm policy add-cluster-role-to-user cluster-admin system:serviceaccount:openshift-infra:template-instance-controller
```

NOTE

- The cluster mode template is only available, by default, on the latest version of the OpenShift Container Platform. It is not provided with the OpenShift Online default catalog.
- The Fuse Console templates configure end-to-end encryption by default so that your Fuse Console requests are secured end-to-end, from the browser to the in-cluster services.
- User management for the Fuse Console is handled by OpenShift.
- Role-based access control (for users accessing the Fuse Console after it is deployed) is not yet available for Fuse on OpenShift.

Section 1.3.1, “Deploying the Fuse Console from the OpenShift 3.11 Console”

Section 1.3.2, “Monitoring a single Fuse pod from the Fuse Console on OpenShift 3.11”

Section 1.3.3, “Deploying the Fuse Console from the command line”

1.3.1. Deploying the Fuse Console from the OpenShift 3.11 Console

To deploy the Fuse Console on your OpenShift cluster from the OpenShift 3.11 Console, follow these steps.

Procedure

1. In the OpenShift console, open an existing project or create a new project.
2. Add the Fuse Console to your OpenShift project:
    a. Select **Add to Project → Browse Catalog**.
    The **Select an item to add to the current project** page opens.
    b. In the **Search** field, type **Fuse Console**.
    The **Red Hat Fuse 7.x Console** and **Red Hat Fuse 7.x Console (cluster)** items should appear as the search result.

> **NOTE**
>
> If the **Red Hat Fuse Console** items do not appear as the search result, or if the items that appear are not the latest version, you can install the Fuse Console templates manually as described in the "Prepare the OpenShift server" section of the Fuse on OpenShift Guide.

a. Click one of the **Red Hat Fuse Console** items:

- **Red Hat Fuse 7.x Console** - This version of the Fuse Console discovers and connects to Fuse applications deployed in the current OpenShift project.
- **Red Hat Fuse 7.x Console (cluster)** - This version of the Fuse Console can discover and connect to Fuse applications deployed across multiple projects on the OpenShift cluster.

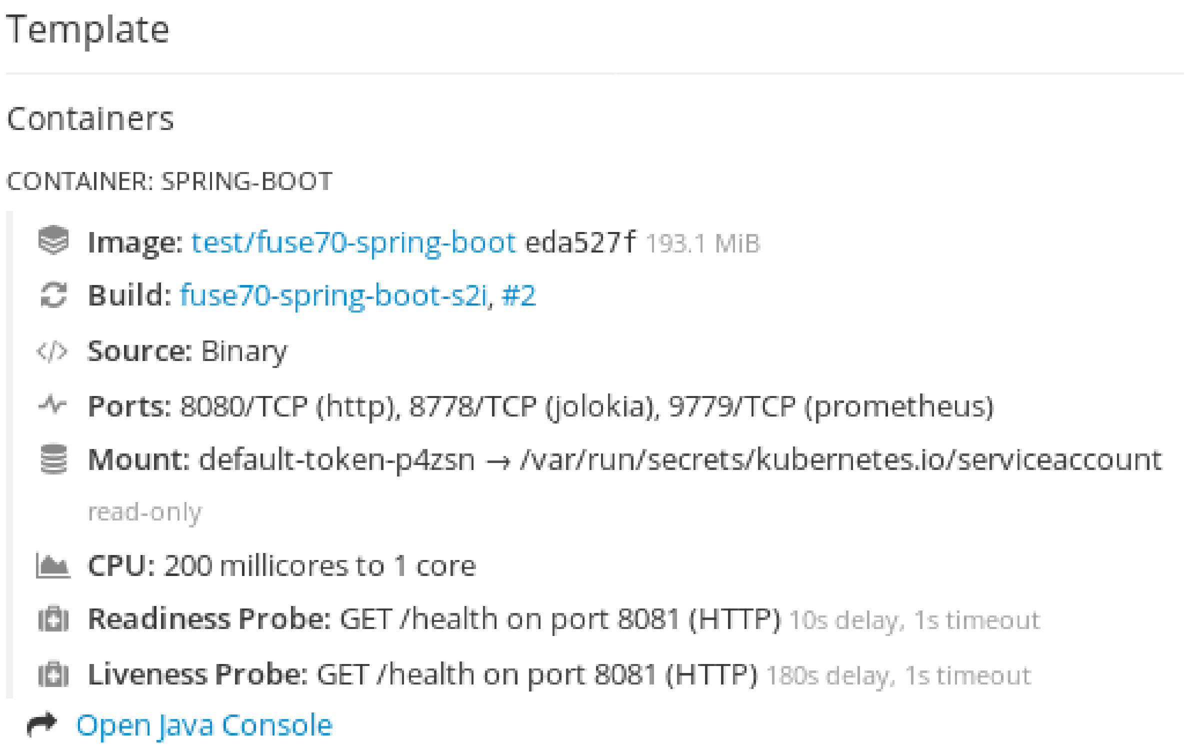
b. In the **Red Hat Fuse Console** wizard, click **Next**. The **Configuration** page of the wizard opens. Optionally, you can change the default values of the configuration parameters.

1. Click **Create**.
The **Results** page of the wizard indicates that the Red Hat Fuse Console has been created.
2. Click the **Continue to the project overview** link to verify that the Fuse Console application is added to the project.
3. To open the Fuse Console, click the provided URL link and then log in.
An **Authorize Access** page opens in the browser listing the required permissions.
4. Click **Allow selected permissions**.
The Fuse Console opens in the browser and shows the Fuse pods running in the project.
5. Click **Connect** for the application that you want to view.
A new browser window opens showing the application in the Fuse Console.

1.3.2. Monitoring a single Fuse pod from the Fuse Console on OpenShift 3.11

You can open the Fuse Console for a Fuse pod running on OpenShift 3.11:

1. From the **Applications → Pods** view in your OpenShift project, click on the pod name to view the details of the running Fuse pod. On the right-hand side of this page, you see a summary of the container template:



Template

Containers

CONTAINER: SPRING-BOOT

Image: test/fuse70-spring-boot eda527f 193.1 MiB

Build: fuse70-spring-boot-s2i, #2

Source: Binary

Ports: 8080/TCP (http), 8778/TCP (jolokia), 9779/TCP (prometheus)

Mount: default-token-p4zsn → /var/run/secrets/kubernetes.io/serviceaccount read-only

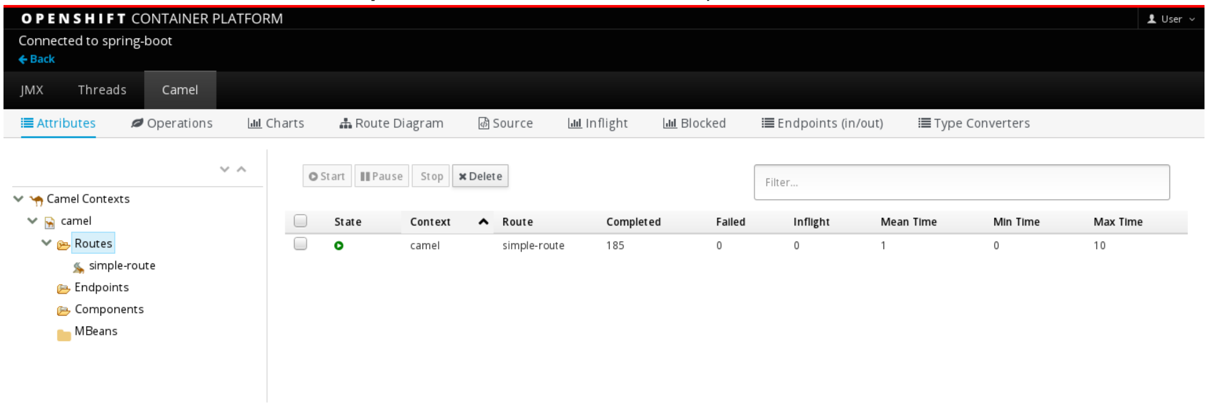
CPU: 200 millicores to 1 core

Readiness Probe: GET /health on port 8081 (HTTP) 10s delay, 1s timeout

Liveness Probe: GET /health on port 8081 (HTTP) 180s delay, 1s timeout

Open Java Console

2. From this view, click on the **Open Java Console** link to open the Fuse Console.



Connected to spring-boot

	State	Context	Route	Completed	Failed	Inflight	Mean Time	Min Time	Max Time
	●	camel	simple-route	185	0	0	1	0	10

NOTE

In order to configure OpenShift to display a link to Fuse Console in the pod view, the pod running a Fuse on OpenShift image must declare a TCP port within a name attribute set to **jolokia**:

```
{
  "kind": "Pod",
  [...]
  "spec": {
    "containers": [
      {
        [...]
        "ports": [
        {
          "name": "jolokia",
          "containerPort": 8778,
          "protocol": "TCP"
        }
```

1.3.3. Deploying the Fuse Console from the command line

Table 1.1, “Fuse Console templates” describes the OpenShift 3.1 templates that you can use to deploy the Fuse Console from the command line, depending on the type of Fuse application deployment.

Table 1.1. Fuse Console templates

Type	Description
fis-console-cluster-template.json	The Fuse Console can discover and connect to Fuse applications deployed across multiple namespaces or projects. To deploy this template, you must have the OpenShift cluster-admin role.
fis-console-namespace-template.json	This template restricts the Fuse Console access to the current OpenShift project (namespace), and as such acts as a single tenant deployment. To deploy this template, you must have the admin role for the current OpenShift project.

Optionally, you can view a list of the parameters for all of the templates by running this command:

```
oc process --parameters -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.0.fuse-760043-redhat-00003/fis-console-namespace-template.json
```

Procedure

To deploy the Fuse Console from the command line:

1. 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 to access the Fuse Console:

     ```
     oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.0.fuse-760043-redhat-00003/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-2.1.0.fuse-760043-redhat-00003/fis-console-namespace-template.json
     ```

NOTE

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

2. Obtain the status and the URL of your Fuse Console deployment by running this command:

```
oc status
```

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

1.4. VIEWING CONTAINERS AND APPLICATIONS

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

Procedure

- To manage (create, edit, or delete) containers, use the OpenShift console.
- To view Fuse applications on the OpenShift cluster, click the **Online** tab.

1.5. VIEWING AND MANAGING APACHE CAMEL APPLICATIONS

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

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

Prerequisite

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

1.5.1. Starting, suspending, or deleting 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:
    a. Stop the context.
    b. Click the ellipse icon and then select **Delete** from the dropdown menu.

NOTE

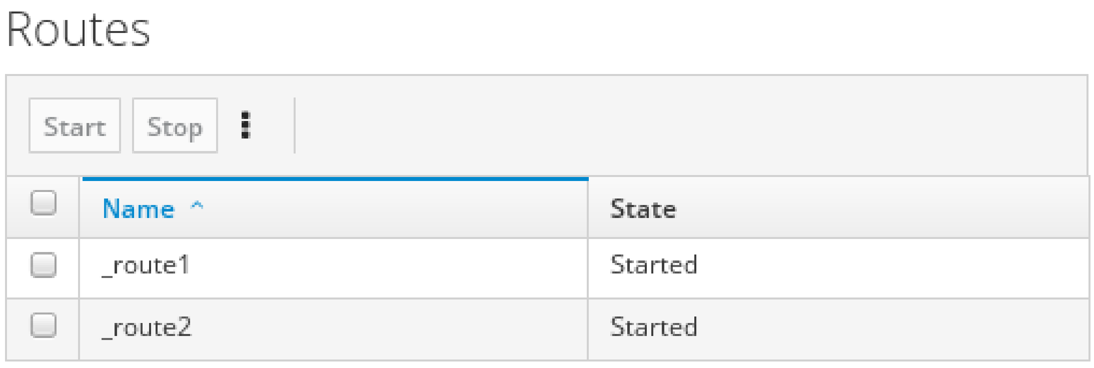
When you delete a context, you remove it from the deployed application.

1.5.2. Viewing 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, click **Chart** and then 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. 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, click **Type Converters**.
7. To view and execute JMX operations, such as adding or updating routes from XML or finding all Camel components available in the classpath, click **Operations**.

1.5.3. Viewing a list of the Camel routes and interacting with them

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



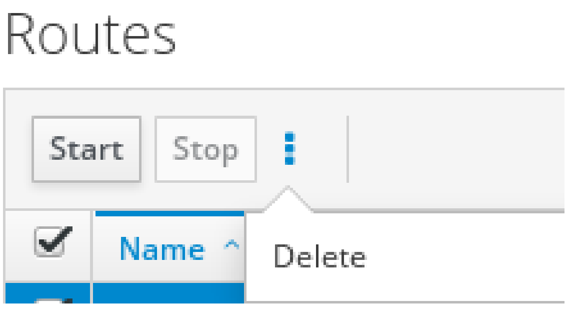
	Name ^	State
☐	_route1	Started
☐	_route2	Started

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.



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.

7. To interact with a specific route:

   a. In the **Camel** tab's tree view, select a route.

   b. To view a list of route attributes and values, click **Attributes**.

   c. 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.

   d. To view inflight and blocked exchanges, click **Exchanges**.

   e. 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.

8. To trace messages through a route:

   a. In the **Camel** tab's tree view, select a route.

   b. Select **Trace**, and then click **Start tracing**.

9. 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**.

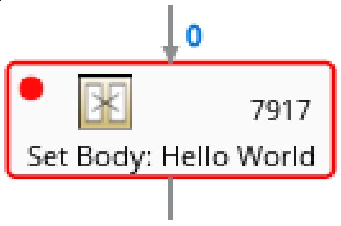
e. Return to the route's **Trace** tab to view the flow of messages through the route.

1.5.4. Debugging a route

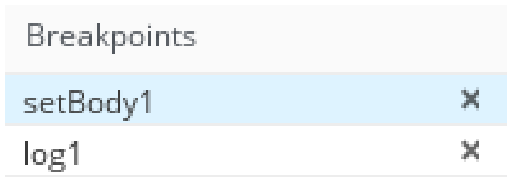
1. In the **Camel** tab's tree view, select a route.

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.

1.6. 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.

Procedure

1. To view and edit MBean attributes:
   a. In the tree view, select an MBean.
   b. Click the **Attributes** tab.
   c. Click an attribute to see its details.
2. To perform operations:
   a. In the tree view, select an MBean.
   b. Click the **Operations** tab, expand one of the listed operations.
   c. Click **Execute** to run the operation.
3. To view charts:
   a. In the tree view, select an item.
   b. Click the **Chart** tab.

1.7. VIEWING DIAGNOSTICS

Use the **Diagnostics** tab to view diagnostic information about the JVM via the JVM DiagnosticCommand and HotspotDiangostic interfaces.

NOTE

The functionality is similar to the **Diagnostic Commands** view in Java Mission Control (jmc) or the command line tool jcmd. The plugin will provide corresponding jcmd commands in some scenarios.

Procedure

1. To retrieve the number of instances of loaded classes and the amount of bytes they take up, click **Class Histogram**. If the operation is repeated, the tab shows the difference since last run.
2. To view the JVM diagnostic flag setting, click the **JVM flags**.
3. For a running JVM, you can also modify the flag settings.

Additional resources

The supported JVM depends on the platform, for more information go to one of the following sources:

- http://www.oracle.com/technetwork/java/vmoptions-jsp-140102.html
- http://openjdk.java.net/groups/hotspot/docs/RuntimeOverview.html

1.8. VIEWING THREADS

You can view and monitor the state of threads.

Procedure

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.

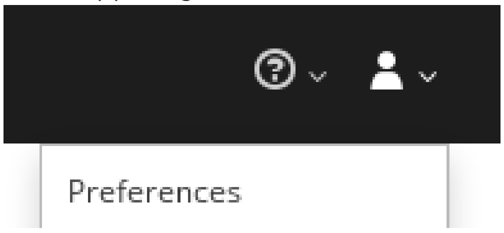
4. To drill down to detailed information for a specific thread, such as the lock class name and full stack trace for that thread, in the **Actions** column, click **More**.

1.9. 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.

Procedure

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**.

CHAPTER 2. MONITORING AND MANAGING RED HAT FUSE APPLICATIONS ON SPRING BOOT STANDALONE

2.1. ABOUT THE FUSE CONSOLE

The Red Hat Fuse Console is a web console based on HawtIO open source software. For a list of supported browsers, go to Supported Configurations.

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

2.2. ACCESSING THE FUSE CONSOLE FOR SPRING BOOT 2.X

You can access the Fuse Console for a standalone Fuse Spring Boot 2.x distribution.

Procedure

1. Add the following dependency to your Fuse application’s **pom.xml** file:

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

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

2. Edit the **src/main/resources/application.properties** file:

   a. Set the following properties:

      - **management.endpoints.web.exposure.include=hawtio,jolokia**
      - **hawtio.authenticationEnabled=false**
      - **management.endpoint.hawtio.enabled=true**
      - **management.endpoint.jolokia.enabled=true**

b. Optionally, set the **management.endpoints.web.base-path** property.
By default for Spring Boot 2.x, the Fuse Console's URL includes the context path (/**actuator**) of the management endpoints. For example:

http://localhost:10001/actuator/hawtio/index.html

To change this default URL, for example to specify the same format as the default Spring Boot 1.x URL (**http://localhost:10001/hawtio**), set the **management.endpoints.web.base-path** property as shown here:

management.endpoints.web.base-path=/

Your **application.properties** settings should look similar to the following example:

```
# ports

server.port=8080

management.server.port=10001

# enable management endpoints for healthchecks and hawtio

management.endpoints.enabled-by-default = false

management.endpoint.hawtio.enabled = true

management.endpoint.jolokia.enabled = true

management.endpoints.health.enabled = true

management.health.defaults.enabled=false

camel.health.enabled=false

camel.health.indicator.enabled=true

management.endpoints.web.exposure.include=hawtio,jolokia

hawtio.authenticationEnabled=false

# change the URL so that it does not include the actuator folder

management.endpoints.web.base-path=/
```

NOTE

By default, authentication for the Fuse Console on Spring Boot is disabled. Optionally, you can enable authentication by writing code specific to your Fuse Console distribution. Here is an example that you can use for guidance:

https://github.com/hawtio/hawtio/tree/master/examples/springboot-authentication

3. Run the Fuse application:

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

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

   ```
   management.server.port   = 10001
   ```

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

   For example, if the **management.server.port** property value is **10001** and you have not set the **management.endpoints.web.base-path** property then the URL is:

   http://localhost:10001/actuator/hawtio/index.html

2.3. ACCESSING THE FUSE CONSOLE FOR SPRING BOOT 1.X

You can access the Fuse Console for a standalone Fuse Spring Boot 1.x distribution.

Procedure

1. Add the following dependency to your Fuse application's **pom.xml** file:

   ```
   <dependency>
     <groupId>io.hawt</groupId>
     <artifactId>hawtio-springboot-1</artifactId>
   </dependency>
   ```

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

2. Edit the **src/main/resources/application.properties** file:

   a. Set the following properties to **false**:

      - **endpoints.jolokia.sensitive**
      - **endpoints.hawtio.sensitive**
      - **hawtio.authenticationEnabled**

   b. Set the following properties to **true**:

      - **endpoints.hawtio.enabled**
      - **endpoints.jolokia.enabled**
        Your **application.properties** settings should look similar to the following example:

        ```
        # ports

        server.port=8080

        management.port=10001
        ```

```
# enable management endpoints for healthchecks and hawtio

endpoints.enabled = false

endpoints.hawtio.enabled = true

endpoints.jolokia.enabled = true

endpoints.health.enabled = true

management.health.defaults.enabled=false

camel.health.enabled=false

camel.health.indicator.enabled=true

endpoints.jolokia.sensitive=false

endpoints.hawtio.sensitive=false

hawtio.authenticationEnabled=false
```

NOTE

By default, authentication for the Fuse Console on Spring Boot is disabled. Optionally, you can enable authentication by writing code specific to your Fuse Console distribution. Here is an example that you can use for guidance:

https://github.com/hawtio/hawtio/tree/master/examples/springboot-authentication

3. Run the Fuse application:

   ```
   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**

2.4. CUSTOMIZING THE FUSE CONSOLE BRANDING

You can customize the Fuse Console branding information, such as title, logo, and login page information, by adding a **hawtconfig.json** file into your Fuse on Spring Boot standalone application.

Procedure

1. Create a JSON file named **hawtconfig.json** in your local Fuse on Spring Boot standalone application's **src/main/webapp** directory.

2. Open the **src/main/webapp/hawtconfig.json** in an editor of your choice, and then add the following content:

```
{
  "branding": {
    "appName": "Red Hat Fuse Console",
    "appLogoUrl": "img/Logo-Red_Hat-Fuse-A-Reverse-RGB.png",
    "companyLogoUrl": "img/Logo-RedHat-A-Reverse-RGB.png"
  },
  "login": {
    "description": "",
    "links": []
  },
  "about": {
    "title": "Red Hat Fuse Console",
    "productInfo": [],
    "additionalInfo": "",
    "copyright": "",
    "imgSrc": "img/Logo-RedHat-A-Reverse-RGB.png"
  },
  "disabledRoutes": [
    "/camel/source",
    "/diagnostics",
    "/jvm/discover",
    "/jvm/local"
  ]
}
```

3. Change the values of the configuration properties listed in Table A.1, "Fuse Console Branding Configuration Properties".

4. Save your changes.

5. Run Fuse on Spring Boot by using the following command:

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

6. In a web browser, open the Fuse Console by using this URL: **http://localhost:10001/actuator/hawtio/index.html**

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.

2.5. SECURING THE FUSE CONSOLE

To secure the Fuse Console on Spring Boot, you can:

- **Set HTTPS as the required protocol**
  You can use the **hawtio.http.strictTransportSecurity** property to require web browsers to use the secure HTTPS protocol to access the Fuse Console. This property specifies that web browsers that try to use HTTP to access the Fuse Console must automatically convert the request to use HTTPS.

- **Use public keys to secure responses**
  You can use the **hawtio.http.publicKeyPins** property to secure the HTTPS protocol by telling the web browser to associate a specific cryptographic public key with the Fuse Console to decrease the risk of "man-in-the-middle" attacks with forged certificates.

Procedure

Set the **hawtio.http.strictTransportSecurity** and **hawtio.http.publicKeyPins** properties as shown in the following example

```
public static void main(String[] args) {
   System.setProperty("hawtio.http.strictTransportSecurity", "max-age=31536000;
includeSubDomains; preload");
   System.setProperty("hawtio.http.publicKeyPins", "pin-
sha256=cUPcTAZWKaASuYWhhneDttWpY3oBAkE3h2+soZS7sWs"; max-age=5184000;
includeSubDomains");
   SpringApplication.run(YourSpringBootApplication.class, args);
}
```

Additional resources

- For a description of the **hawtio.http.strictTransportSecurity** property's syntax, see the description page for the HTTP Strict Transport Security (HSTS) response header.

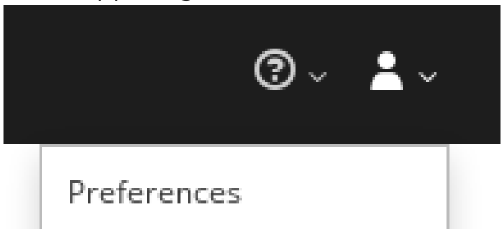
- For a description of the **hawtio.http.publicKeyPins** property's syntax, including instructions on how to extract the Base64 encoded public key, see the description page for the HTTP Public Key Pinning response header.

2.6. 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.

Procedure

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**.

2.7. CONNECTING TO REMOTE FUSE APPLICATIONS

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).

Procedure

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 **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");
```

2.7.1. Connecting to a remote Jolokia agent

Before you begin, you need to know the connection details (host name, port, and path) of the remote Jolokia agent.

The default connection URLs for the Jolokia agent on Spring Boot is **http://<host>:8080/jolokia**

As a system administrator, you can change this default.

Typically, the URL to remotely connect to a Jolokia agent is the URL to open the Fuse Console plus **/jolokia**. For example, if the URL to open the Fuse Console is **http://<host>:1234/hawtio**, then the URL to remotely connect to it would probably be **http://<host>:1234/hawtio/jolokia**.

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**.
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.

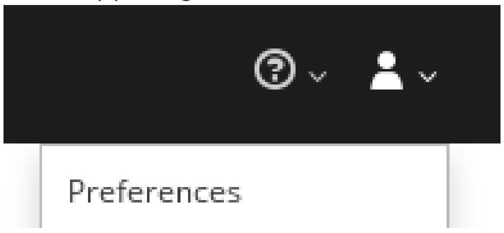
2.7.2. Setting data moving preferences

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**.

2.7.3. Viewing JVM runtime information

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

2.8. VIEWING AND MANAGING APACHE CAMEL APPLICATIONS

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

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

Prerequisite

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

2.8.1. Starting, suspending, or deleting 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:
    a. Stop the context.
    b. Click the ellipse icon and then select **Delete** from the dropdown menu.

NOTE

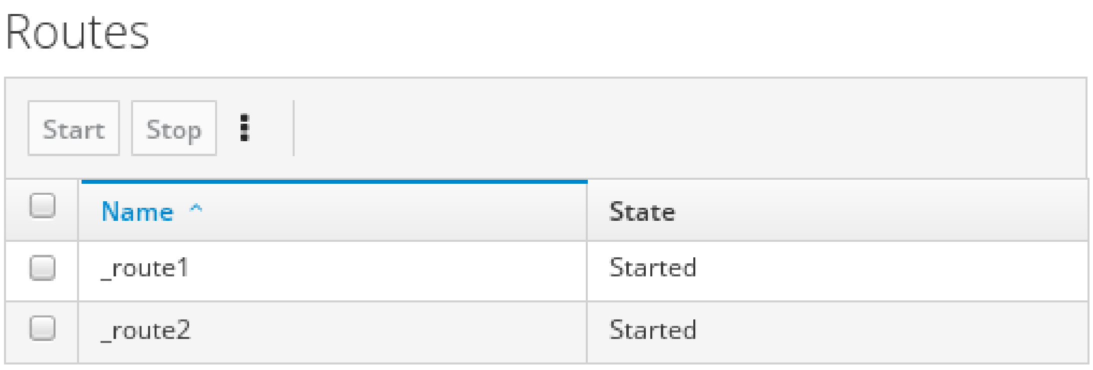
When you delete a context, you remove it from the deployed application.

2.8.2. Viewing 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, click **Chart** and then 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. 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, click **Type Converters**.
7. To view and execute JMX operations, such as adding or updating routes from XML or finding all Camel components available in the classpath, click **Operations**.

2.8.3. Viewing a list of the Camel routes and interacting with them

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



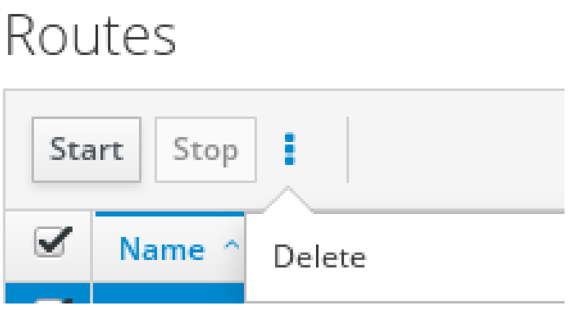
	Name ^	State
☐	_route1	Started
☐	_route2	Started

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.



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.

7. To interact with a specific route:

    a. In the **Camel** tab's tree view, select a route.

    b. To view a list of route attributes and values, click **Attributes**.

    c. 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.

d. To view inflight and blocked exchanges, click **Exchanges**.

e. 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.

8. To trace messages through a route:

   a. In the **Camel** tab's tree view, select a route.

   b. Select **Trace**, and then click **Start tracing**.

9. 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**.

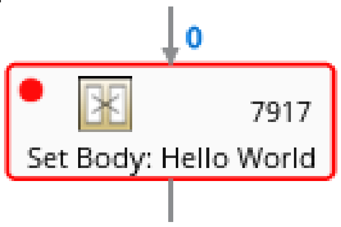
   e. Return to the route's **Trace** tab to view the flow of messages through the route.

2.8.4. Debugging a route

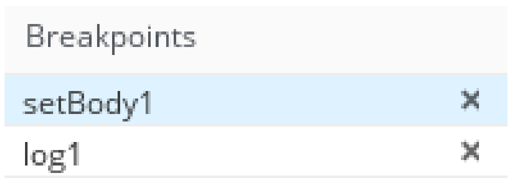
1. In the **Camel** tab's tree view, select a route.

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.

2.9. 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.

Procedure

1. To view and edit MBean attributes:
   a. In the tree view, select an MBean.
   b. Click the **Attributes** tab.
   c. Click an attribute to see its details.
2. To perform operations:
   a. In the tree view, select an MBean.
   b. Click the **Operations** tab, expand one of the listed operations.
   c. Click **Execute** to run the operation.
3. To view charts:
   a. In the tree view, select an item.
   b. Click the **Chart** tab.

2.10. VIEWING DIAGNOSTICS

Use the **Diagnostics** tab to view diagnostic information about the JVM via the JVM DiagnosticCommand and HotspotDiangostic interfaces.

NOTE

The functionality is similar to the **Diagnostic Commands** view in Java Mission Control (jmc) or the command line tool jcmd. The plugin will provide corresponding jcmd commands in some scenarios.

Procedure

1. To retrieve the number of instances of loaded classes and the amount of bytes they take up, click **Class Histogram**. If the operation is repeated, the tab shows the difference since last run.
2. To view the JVM diagnostic flag setting, click the **JVM flags**.
3. For a running JVM, you can also modify the flag settings.

Additional resources

The supported JVM depends on the platform, for more information go to one of the following sources:

- http://www.oracle.com/technetwork/java/vmoptions-jsp-140102.html
- http://openjdk.java.net/groups/hotspot/docs/RuntimeOverview.html

2.11. VIEWING THREADS

You can view and monitor the state of threads.

Procedure

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 3. MONITORING AND MANAGING RED HAT FUSE APPLICATIONS ON KARAF STANDALONE

3.1. ABOUT THE FUSE CONSOLE

The Red Hat Fuse Console is a web console based on HawtIO open source software. For a list of supported browsers, go to Supported Configurations.

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

3.2. ACCESSING THE FUSE CONSOLE

To access the Fuse Console for Apache Karaf standalone, follow these steps.

Prerequisite

Install Fuse on the Karaf container. For step-by-step instructions, see Installing on Apache Karaf.

Procedure

1. 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.

2. In a browser, type the URL to connect to the Fuse Console. For example: **http://localhost:8181/hawtio**

3. In the login page, type 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.

3.3. SECURING THE FUSE CONSOLE

To secure the Fuse Console on Apache Karaf, you can:

- **Set HTTPS as the required protocol**
  You can use the **hawtio.http.strictTransportSecurity** property to require web browsers to use the secure HTTPS protocol to access the Fuse Console. This property specifies that web browsers that try to use HTTP to access the Fuse Console must automatically convert the request to use HTTPS.

- **Use public keys to secure responses**
  You can use the **hawtio.http.publicKeyPins** property to secure the HTTPS protocol by telling the web browser to associate a specific cryptographic public key with the Fuse Console to decrease the risk of “man-in-the-middle” attacks with forged certificates.

- **Enable 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.

- **Implement Red Hat Single Sign On**

- **Control user access**
  The operations that an authenticated user is allowed to perform depend on the role (or roles) assigned to that user, as listed in Table 3.1, “Role-based access on Karaf standalone” .

Procedure

1. To set HTTPS as the required protocol, set the **hawtio.http.strictTransportSecurity** property in the **$KARAF_HOME/etc/system.properties** file as shown in the following example:

   ```
   hawtio.http.strictTransportSecurity = max-age=31536000; includeSubDomains; preload
   ```

2. To use public keys to secure responses, set the **hawtio.http.publicKeyPins** property in the **$KARAF_HOME/etc/system.properties** file as shown in the following example:

   ```
   hawtio.http.publicKeyPins = pin-
   sha256="cUPcTAZWKaASuYWhhneDttWpY3oBAkE3h2+soZS7sWs"; max-age=5184000;
   includeSubDomains
   ```

3. For detailed instructions on how to enable SSL/TLS security, see the "Enabling SSL/TLS for Undertow in an Apache Karaf container" section in the Apache Karaf Security Guide .

4. For information on how to secure the Fuse Console with Red Hat Single Sign-On, see the section on securing the Hawtio administration console in the *Red Hat Single Sign-on Securing Applications and Services Guide*.

5. To ensure that a user has the necessary user role authorization to perform the Fuse Console operations that the user needs to perform, follow these steps to set a user role:

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

   b. 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
```

c. Save the file.

Additional resources

- For a description of the **hawtio.http.strictTransportSecurity** property's syntax, see the description page for the HTTP Strict Transport Security (HSTS) response header.
- For a description of the **hawtio.http.publicKeyPins** property's syntax, including instructions on how to extract the Base64 encoded public key, see the description page for the HTTP Public Key Pinning response header.

3.4. ROLE-BASED ACCESS REFERENCE

The operations that an authenticated user is allowed to perform depend on the role (or roles) assigned to that user, as listed in Table 3.1, "Role-based access on Karaf standalone".

Table 3.1. Role-based access on Karaf standalone

Operation	admin	manager	viewer
Log in/Log out	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	
Send messages	Y	Y	
Add endpoints	Y	Y	

Operation	admin	manager	viewer
View routes, route diagrams, and runtime statistics	Y	Y	Y
Start and stop routes	Y	Y	
Delete routes	Y	Y	
JMX			
Change attribute values	Y	Y	
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	
Add configurations	Y	Y	
Install and uninstall features	Y		
Runtime			
View system properties, metrics, and threads	Y	Y	Y
Logs			
View logs	Y	Y	Y

Additional resources

For more information on role-based access control, see Deploying into Apache Karaf.

3.5. CUSTOMIZING THE FUSE CONSOLE BRANDING

You can customize the Fuse Console branding information, such as title, logo, and login page information, by using the Fuse Console branding plugin.

By default, the Fuse Console branding is defined in the **hawtconfig.json** that is located in the Fuse Console WAR file (**karaf-install-dir/system/io/hawt/hawtio-war/<version>/hawtio-war-<version>.war**). When you implement the Fuse Console branding plugin, you can override the default branding with your own custom branding.

Procedure

1. Download the branding plugin example from **https://github.com/hawtio/hawtio/tree/hawtio-2.9.1/examples/branding-plugin** to a local directory of your choice.

2. In an editor of your choice, open the Fuse Console branding plugin's **src/main/webapp/plugin/brandingPlugin.js** file to customize the Fuse Console branding. You can change the values of the configuration properties listed in Table A.1, "Fuse Console Branding Configuration Properties".

3. Save your changes.

4. In an editor of your choice, open the Fuse Console branding plugin's **pom.xml** file to its **<parent>** section:

```
<parent>
    <groupId>io.hawt</groupId>
    <artifactId>project</artifactId>
    <version>2.9-SNAPSHOT</version>
    <relativePath>../..</relativePath>
  </parent>
```

5. Edit the **<parent>** section as follows:

    a. Change the value of the **<version>** property to match the version of your Fuse on Karaf installation. For example, if your Fuse on Karaf installation directory name is 2.0.0.fuse-760015, set the version to 2.0.0.fuse-760015.

    b. Remove the **<relativePath>../..</relativePath>** line.
    For example:

    ```
    <parent>
        <groupId>io.hawt</groupId>
        <artifactId>project</artifactId>
        <version> 2.0.0.fuse-760015</version>
          </parent>
    ```

6. In a Terminal window, build the branding-plugin project by running the following command:

```
mvn clean install
```

7. If Fuse is not already running, start it by running the following command:
Linux/Unix: bin/fuse

Windows: bin\fuse.bat`

8. At the Karaf CLI prompt, type the following command to install the Fuse Console branding plugin (where **<version>** is the version of your Fuse on Karaf installation):
   Linux/Unix: **install -s mvn:io.hawt/branding-plugin/<version>/war**

   Windows: **install -s mvn:io.hawt\branding-plugin\<version>\war**

9. In a web browser, open the Fuse Console by using the URL that the start command returned in Step 7 (the default URL is http://localhost:8181/hawtio/).

NOTE

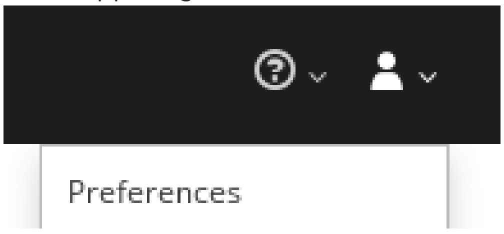
If you have already run the Fuse Console in a web browser, the branding is stored in the browser's local storage. To use new branding settings, you must clear the browser's local storage.

3.6. 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.

Procedure

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**.

3.7. DISABLING THE FUSE CONSOLE

You can disable the Fuse Console on Karaf so that it becomes inaccessible to all users without affecting any other component.

Procedure

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.

Additional resources

For more information about managing bundles, see the "Lifecycle Management" chapter of Deploying into Apache Karaf.

3.8. CONNECTING TO REMOTE FUSE APPLICATIONS

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).

3.8.1. Unlocking the Fuse Console

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

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

1. Open the **$KARAF_HOME/etc/jolokia-access.xml** file in an editor.

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.

3.8.2. Restricting remote access

Optionally, you can restrict remote access to the Fuse Console for specific hosts and IP addresses.

You can grant overall access based on the IP address of an HTTP client. To specify these restrictions:

In the **jolokia-access.xml** file, add or edit a **<remote>** section that contains one or more **<host>** elements. For the **<host>** element, you can specify an IP address, a host name, or a netmask given in CIDR format (for example, **10.0.0.0/16** for all clients coming from the 10.0 network).

The following example allows access from localhost and all clients whose IP addresses start with **10.0**. For all other IP addresses, access is denied.

```
<remote>
  <host>localhost</host>
  <host>10.0.0.0/16</host>
</remote>
```

For more details, see the Jolokia security documentation (https://jolokia.org/reference/html/security.html).

3.8.3. 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
```

3.8.4. Connecting to a remote Jolokia agent

Before you begin, you need to know the connection details (host name, port, and path) of the remote Jolokia agent.

The default connection URL for the Jolokia agent for Fuse on Apache Karaf is **http://<host>:8181/hawtio/jolokia**.

As a system administrator, you can change this default.

Typically, the URL to remotely connect to a Jolokia agent is the URL to open the Fuse Console plus **/jolokia**. For example, if the URL to open the Fuse Console is **http://<host>:1234/hawtio**, then the URL to remotely connect to it would probably be **http://<host>:1234/hawtio/jolokia**.

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**.
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.

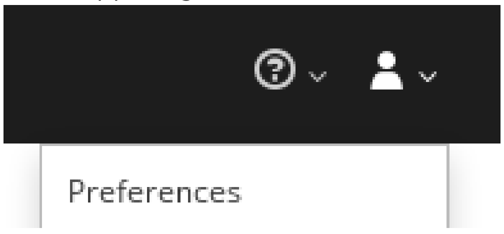
3.8.5. Setting data moving preferences

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**.

3.8.6. Viewing JVM runtime information

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

3.9. VIEWING AND MANAGING APACHE CAMEL APPLICATIONS

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

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

Prerequisite

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

3.9.1. Starting, suspending, or deleting 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:
   a. Stop the context.
   b. Click the ellipse icon and then select **Delete** from the dropdown menu.

NOTE

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

3.9.2. Viewing 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, click **Chart** and then 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. 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, click **Type Converters**.

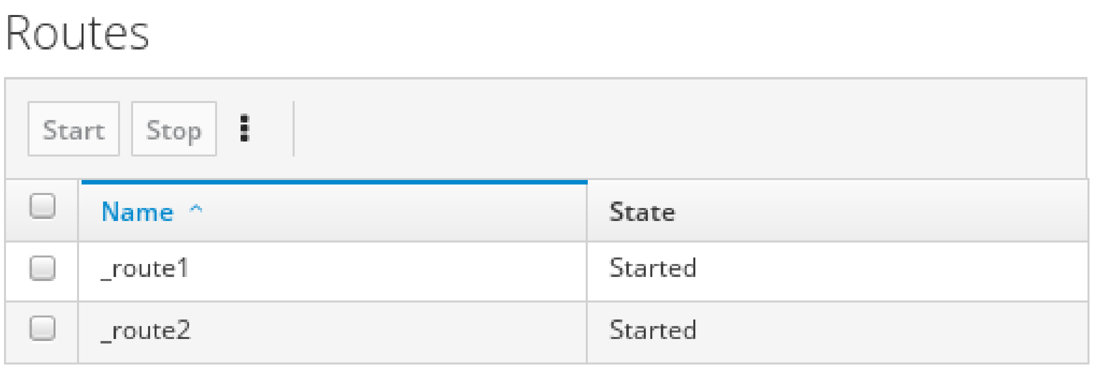
7. To view and execute JMX operations, such as adding or updating routes from XML or finding all Camel components available in the classpath, click **Operations**.

3.9.3. Viewing a list of the Camel routes and interacting with them

1. To view a list of routes:

   a. Click the **Camel** tab.

   b. In the tree view, click the application's routes folder:



Routes

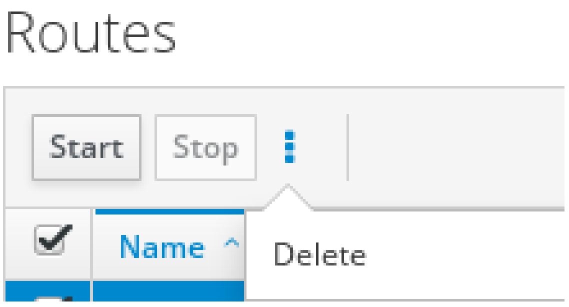
	Name ^	State
☐	_route1	Started
☐	_route2	Started

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.



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.

7. To interact with a specific route:

   a. In the **Camel** tab's tree view, select a route.

   b. To view a list of route attributes and values, click **Attributes**.

   c. 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.

   d. To view inflight and blocked exchanges, click **Exchanges**.

   e. 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.

8. To trace messages through a route:

   a. In the **Camel** tab's tree view, select a route.

   b. Select **Trace**, and then click **Start tracing**.

9. 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**.

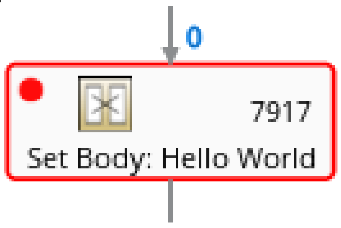
   e. Return to the route's **Trace** tab to view the flow of messages through the route.

3.9.4. Debugging a route

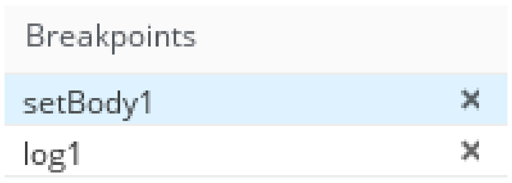
1. In the **Camel** tab's tree view, select a route.

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:



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.

3.10. 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.

Procedure

1. To view and edit MBean attributes:
   a. In the tree view, select an MBean.
   b. Click the **Attributes** tab.
   c. Click an attribute to see its details.
2. To perform operations:
   a. In the tree view, select an MBean.
   b. Click the **Operations** tab, expand one of the listed operations.
   c. Click **Execute** to run the operation.
3. To view charts:
   a. In the tree view, select an item.
   b. Click the **Chart** tab.

3.11. VIEWING AND MANAGING YOUR OSGI ENVIRONMENT

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.

3.12. VIEWING DIAGNOSTICS

Use the **Diagnostics** tab to view diagnostic information about the JVM via the JVM DiagnosticCommand and HotspotDiangostic interfaces.

NOTE

The functionality is similar to the **Diagnostic Commands** view in Java Mission Control (jmc) or the command line tool jcmd. The plugin will provide corresponding jcmd commands in some scenarios.

Procedure

1. To retrieve the number of instances of loaded classes and the amount of bytes they take up, click **Class Histogram**. If the operation is repeated, the tab shows the difference since last run.

2. To view the JVM diagnostic flag setting, click the **JVM flags**.

3. For a running JVM, you can also modify the flag settings.

Additional resources

The supported JVM depends on the platform, for more information go to one of the following sources:

- http://www.oracle.com/technetwork/java/vmoptions-jsp-140102.html

- http://openjdk.java.net/groups/hotspot/docs/RuntimeOverview.html

3.13. VIEWING THREADS

You can view and monitor the state of threads.

Procedure

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**.

3.14. VIEWING LOG ENTRIES

You can view log entries for Red Hat Fuse in the **Logs** tab.

Prerequisite

The **Logs** tab is available when the Java application includes the Log MBean.

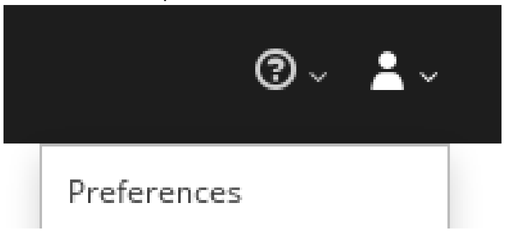
Procedure

1. To view a list of the log entries, click the **Log Entries** tab.
   By default, the list shows log entries in ascending order.

   You can drill down to each log entry to view detailed information about the log entry.

2. To filter the list of logs to show specific log types, click the **Action Bar**. You can filter the log entries section according to a text string or the logging level.

3. To change the Fuse Console default settings:

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

   b. To change the default sorting order, select **Server Logs** and then 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.

   c. Optionally, you can customize these 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 messages to include, such as **hawtio-oauth** and **hawtio-core-utils**.

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

CHAPTER 4. MONITORING AND MANAGING RED HAT FUSE APPLICATIONS ON EAP STANDALONE

4.1. ABOUT THE FUSE CONSOLE

The Red Hat Fuse Console is a web console based on HawtIO open source software. For a list of supported browsers, go to Supported Configurations.

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

4.2. ACCESSING THE FUSE CONSOLE

Follow these steps to access the Fuse Console for Red Hat JBoss Enterprise Application Platform.

Prerequisite

You must install Fuse on the JBoss EAP container. For step-by-step instructions, see Installing on JBoss EAP.

Procedure

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.

NOTE

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.

4.3. CUSTOMIZING THE FUSE CONSOLE BRANDING

You can customize the Fuse Console branding information, such as title, logo, and login page information, by using the Fuse Console branding plugin.

By default, the Fuse Console branding is defined in the **hawtconfig.json** that is located in the Fuse Console WAR file (**eap-install-dir/standalone/deployments/hawtio-wildfly-<version>.war**). When you implement the Fuse Console branding plugin, you can override the default branding with your own custom branding.

Procedure

1. Download the branding plugin example from **https://github.com/hawtio/hawtio/tree/master/examples/branding-plugin** to a local directory of your choice.

2. In an editor of your choice, open the Fuse Console branding plugin's **src/main/webapp/plugin/brandingPlugin.js** file to customize the Fuse Console branding. You can change the values of the configuration properties listed in Table A.1, "Fuse Console Branding Configuration Properties".

3. Save your changes.

4. In an editor of your choice, open the Fuse Console branding plugin's **pom.xml** file to its **<parent>** section:

```
<parent>
   <groupId>io.hawt</groupId>
   <artifactId>project</artifactId>
   <version>2.9-SNAPSHOT</version>
   <relativePath>../..</relativePath>
 </parent>
```

5. Edit the **<parent>** section as follows:

   a. Change the value of the **<version>** property to match the version of your Fuse on EAP installation. For example, if your Fuse on EAP installation directory name is **2.0.0.fuse-760015**, set the version to **2.0.0.fuse-760015**.

   b. Remove the **<relativePath>../..</relativePath>** line.
      For example:

```
<parent>
   <groupId>io.hawt</groupId>
   <artifactId>project</artifactId>
   <version> 2.0.0.fuse-760015</version>
     </parent>
```

6. In a Terminal window, build the branding-plugin project by running the following command:

   ```
   mvn clean install
   ```

   This command creates a **branding-plugin.war** file in the project's **/target** folder.

7. Copy the **branding-plugin.war** file to your EAP installation's **standalone/deployments** directory.

8. If Fuse is not already running, start it by running the following command:
   On Linux/Mac OS: **./bin/standalone.sh**

   On Windows: **./bin/standalone.bat**

9. In a web browser, open the Fuse Console by using the URL that the start command returned in the previous step (the default URL is **http://localhost:8080/hawtio**).

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.

4.4. SECURING THE FUSE CONSOLE

To secure the Fuse Console on EAP, you can:

- **Set HTTPS as the required protocol**
  You can use the **hawtio.http.strictTransportSecurity** property to require web browsers to use the secure HTTPS protocol to access the Fuse Console. This property specifies that web browsers that try to use HTTP to access the Fuse Console must automatically convert the request to use HTTPS.

- **Use public keys to secure responses**
  You can use the **hawtio.http.publicKeyPins** property to secure the HTTPS protocol by telling the web browser to associate a specific cryptographic public key with the Fuse Console to decrease the risk of “man-in-the-middle” attacks with forged certificates.

Procedure

Set the **hawtio.http.strictTransportSecurity** and the **hawtio.http.publicKeyPins** properties in the **system-properties** section of the **$EAP_HOME/standalone/configuration/standalone*.xml** file as shown in the following example:

```
<property name="hawtio.http.strictTransportSecurity" value="max-age=31536000;
includeSubDomains; preload"/>
<property name="hawtio.http.publicKeyPins" value="pin-
sha256=cUPcTAZWKaASuYWhhneDttWpY3oBAkE3h2+soZS7sWs"; max-age=5184000;
includeSubDomains"/>
```

Additional resources

- For a description of the **hawtio.http.strictTransportSecurity** property's syntax, see the description page for the HTTP Strict Transport Security (HSTS) response header.

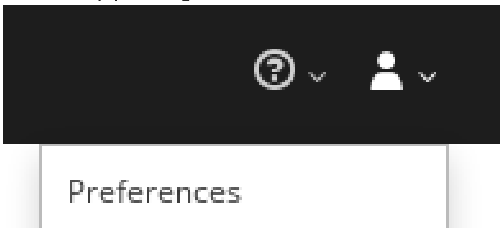
- For a description of the **hawtio.http.publicKeyPins** property's syntax, including instructions on how to extract the Base64 encoded public key, see the description page for the HTTP Public Key Pinning response header.

4.5. 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.

Procedure

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**.

4.6. DISABLING THE FUSE CONSOLE

You can disable the Fuse Console on JBoss EAP so that it becomes inaccessible to all users without affecting any other component.

Procedure

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

- Remove the Fuse Console 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.

4.7. CONNECTING TO REMOTE FUSE APPLICATIONS

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).

4.7.1. Unlocking the Fuse Console

By default, Jolokia for Fuse 7 standalone on 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 **locahost** or **127.0.0.1**, follow these steps:

1. Open the **$EAP_HOME/standalone/configuration/jolokia-access.xml** file in an editor.
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.

4.7.2. Restricting remote access

Optionally, you can restrict remote access to the Fuse Console for specific hosts and IP addresses.

You can grant overall access based on the IP address of an HTTP client. To specify these restrictions:

In the **jolokia-access.xml** file, add or edit a **<remote>** section that contains one or more **<host>** elements. For the **<host>** element, you can specify an IP address, a host name, or a netmask given in CIDR format (for example, **10.0.0.0/16** for all clients coming from the 10.0 network).

The following example allows access from localhost and all clients whose IP addresses start with **10.0**. For all other IP addresses, access is denied.

```
<remote>
  <host>localhost</host>
  <host>10.0.0.0/16</host>
</remote>
```

For more details, see the Jolokia security documentation (https://jolokia.org/reference/html/security.html).

4.7.3. 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, 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"/>
```

4.7.4. Connecting to a remote Jolokia agent

Before you begin, you need to know the connection details (host name, port, and path) of the remote Jolokia agent.

The default connection URLs for the Jolokia agent on Red Hat JBoss EAP is **http://<host>:8080/hawtio/jolokia**.

As a system administrator, you can change this default.

Typically, the URL to remotely connect to a Jolokia agent is the URL to open the Fuse Console plus **/jolokia**. For example, if the URL to open the Fuse Console is **http://<host>:1234/hawtio**, then the URL to remotely connect to it would probably be **http://<host>:1234/hawtio/jolokia**.

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

NOTE

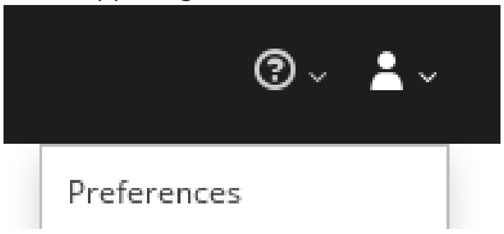
The Fuse Console automatically probes the local network interfaces other than localhost and 127.0.0.1 and adds them to the whitelist. Hence, you do not need to manually register the local machine's addresses to the whitelist.

4.7.5. 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**.

4.7.6. Viewing JVM runtime information

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

4.8. VIEWING AND MANAGING APACHE CAMEL APPLICATIONS

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

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

Prerequisite

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

4.8.1. Starting, suspending, or deleting 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:
   a. Stop the context.
   b. Click the ellipse icon and then select **Delete** from the dropdown menu.

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

4.8.2. Viewing Camel application details

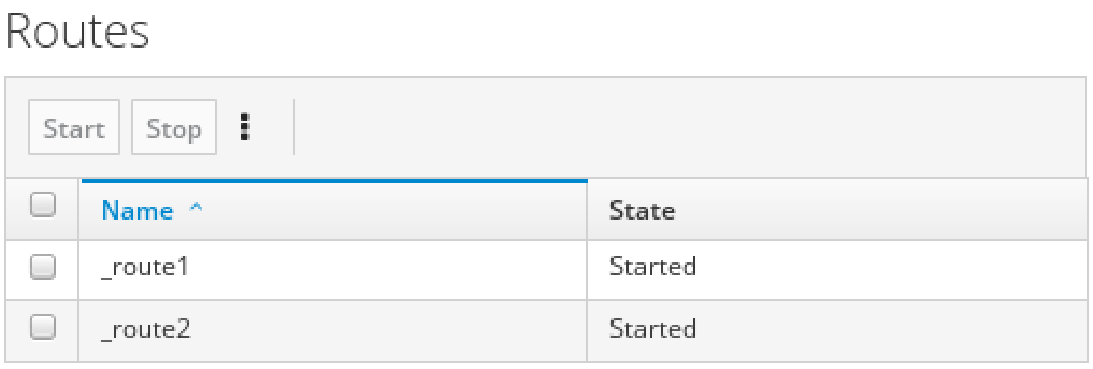
1. In the **Camel** tab's tree view, click a Camel application.
2. To view a list of application attributes and values, click **Attributes**.
3. To view a graphical representation of the application attributes, click **Chart** and then click **Edit** to select the attributes that you want to see in the 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. 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, click **Type Converters**.
7. To view and execute JMX operations, such as adding or updating routes from XML or finding all Camel components available in the classpath, click **Operations**.

4.8.3. Viewing a list of the Camel routes and interacting with them

1. To view a list of routes:

a. Click the **Camel** tab.

b. In the tree view, click the application's routes folder:



Routes

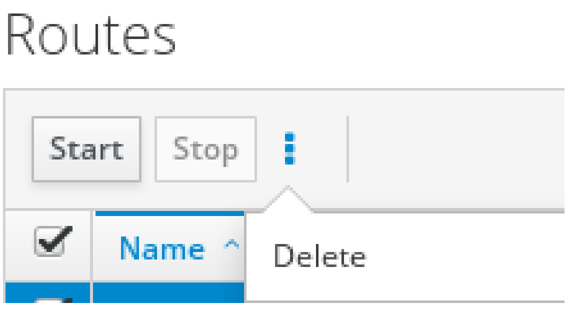
	Name ^	State
☐	_route1	Started
☐	_route2	Started

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.



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.

7. To interact with a specific route:

   a. In the **Camel** tab's tree view, select a route.

   b. To view a list of route attributes and values, click **Attributes**.

c. 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.

d. To view inflight and blocked exchanges, click **Exchanges**.

e. 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.

8. To trace messages through a route:

   a. In the **Camel** tab's tree view, select a route.

   b. Select **Trace**, and then click **Start tracing**.

9. 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**.

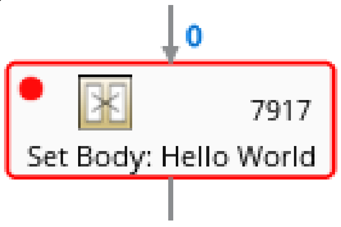
   e. Return to the route's **Trace** tab to view the flow of messages through the route.

4.8.4. Debugging a route

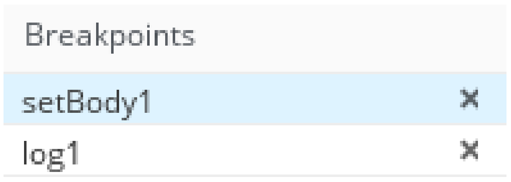
1. In the **Camel** tab's tree view, select a route.

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:



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.

4.9. 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.

Procedure

1. To view and edit MBean attributes:
    a. In the tree view, select an MBean.
    b. Click the **Attributes** tab.
    c. Click an attribute to see its details.
2. To perform operations:
    a. In the tree view, select an MBean.
    b. Click the **Operations** tab, expand one of the listed operations.
    c. Click **Execute** to run the operation.
3. To view charts:
    a. In the tree view, select an item.
    b. Click the **Chart** tab.

4.10. VIEWING DIAGNOSTICS

Use the **Diagnostics** tab to view diagnostic information about the JVM via the JVM DiagnosticCommand and HotspotDiangostic interfaces.

NOTE

The functionality is similar to the **Diagnostic Commands** view in Java Mission Control (jmc) or the command line tool jcmd. The plugin will provide corresponding jcmd commands in some scenarios.

Procedure

1. To retrieve the number of instances of loaded classes and the amount of bytes they take up, click **Class Histogram**. If the operation is repeated, the tab shows the difference since last run.
2. To view the JVM diagnostic flag setting, click the **JVM flags**.
3. For a running JVM, you can also modify the flag settings.

Additional resources

The supported JVM depends on the platform, for more information go to one of the following sources:

- http://www.oracle.com/technetwork/java/vmoptions-jsp-140102.html
- http://openjdk.java.net/groups/hotspot/docs/RuntimeOverview.html

4.11. VIEWING THREADS

You can view and monitor the state of threads.

Procedure

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**.

4.12. VIEWING LOG ENTRIES

You can view log entries for Red Hat Fuse in the **Logs** tab.

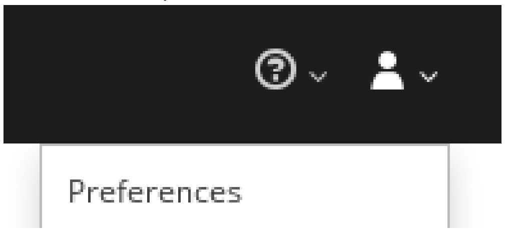
Prerequisite

The **Logs** tab is available when the Java application includes the Log MBean.

Procedure

1. To view a list of the log entries, click the **Log Entries** tab.
   By default, the list shows log entries in ascending order.

   You can drill down to each log entry to view detailed information about the log entry.
2. To filter the list of logs to show specific log types, click the **Action Bar**. You can filter the log entries section according to a text string or the logging level.
3. To change the Fuse Console default settings:
   a. In the upper right corner of the Fuse Console, click the user icon and then click **Preferences** from the drop-down menu.



b. To change the default sorting order, select **Server Logs** and then 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.

c. Optionally, you can customize these 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 messages to include, such as **hawtio-oauth** and **hawtio-core-utils**.

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

APPENDIX A. FUSE CONSOLE BRANDING CONFIGURATION PROPERTIES

By default, the Fuse Console branding is defined in the **hawtconfig.json** file that is located in the Fuse Console WAR file. You can customize the Fuse Console branding information, such as title, logo, and login page information, by using the Fuse Console branding plugin.

Table A.1, "Fuse Console Branding Configuration Properties" provides a description of the properties and lists whether or not each property requires a value.

Table A.1. Fuse Console Branding Configuration Properties

Section	Property Name	Value in Default **hawtconfig.json** file	Description	Value Required?
branding	appName	Red Hat Fuse Management Console	The name for your application. This name displays in the title bar of the Fuse Console.	Required
	appLogoUrl	**img/Logo-Red_Hat-Fuse-A-Reverse-RGB.png**	The path to your application log image file.	Required
	companyLogoUrl	**img/Logo-RedHat-A-Reverse-RGB.png**	The path to your company logo image file.	Required
login	description	*Empty value*	Descriptive text that displays on the Fuse Console Login page (for example, **http://localhost:8181/hawtio**).	Optional
	links	[ ]	Specify an array of **"url"** and **"text"** pairs to provide additional links to pages where the user can get more information or help.	Optional

Section	Property Name	Value in Default **hawtconfig.json** file	Description	Value Required?
about	title	Red Hat Fuse Management Console	The title that shows on the About page of the Fuse Console.	Required
	productInfo	*Empty value*	Product information that shows on the About page of the Fuse Console.	Optional
	additionalInfo	*Empty value*	Any additional information that shows on the About page of the Fuse Console.	Optional
	copyright	*Empty value*	Copyright information that shows on the About page of the Fuse Console.	Optional
	imgSrc	**img/Logo-RedHat-A-Reverse-RGB.png**	The image that appears on the About page of the Fuse Console.	Required
disabledRoutes	*none*	[ ]	Disables specific paths (i.e., plugins) on the console. Do not change this section. Any change is not supported.	Required