Red Hat OpenShift Application Runtimes 1

Thorntail Runtime Guide

For Use with Red Hat OpenShift Application Runtimes

Last Updated: 2018-10-24

Red Hat OpenShift Application Runtimes 1 Thorntail Runtime Guide

For Use with Red Hat OpenShift Application Runtimes

Legal Notice

Abstract

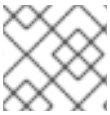
This guide provides details on using the Thorntail runtime with Red Hat OpenShift Application Runtimes.

Table of Contents

PREFACE

This guide covers concepts as well as practical details needed by developers to use the Thorntail runtime.

CHAPTER 1. WHAT IS THORNTAIL



NOTE

Thorntail was formerly known as WildFly Swarm.

Thorntail deconstructs the features in JBoss EAP and allows them to be selectively reconstructed based on the needs of your application. This allows you to create microservices that run on a *just-enough-appserver* that supports the exact subset of APIs you need. See Additional Resources for further information about Thorntail.

The Thorntail runtime enables you to run Thorntail applications and services in OpenShift while providing all the advantages and conveniences of the OpenShift platform such as rolling updates, service discovery, and canary deployments. OpenShift also makes it easier for your applications to implement common microservice patterns such as externalized configuration, health check, circuit breaker, and failover.

Thorntail has a product version of its runtime that runs on OpenShift and is provided as part of a Red Hat subscription.

CHAPTER 2. MISSIONS AND CLOUD-NATIVE DEVELOPMENT ON OPENSHIFT

When developing applications on OpenShift, you can use missions and boosters to kickstart your development.

Missions

Missions are working applications that showcase different fundamental pieces of building cloud native applications and services.

A mission implements a Microservice pattern such as:

- Creating REST APIs
- Interoperating with a database
- Implementing the Health Check pattern

You can use missions for a variety of purposes:

- A proof of technology demonstration
- A teaching tool, or a sandbox for understanding how to develop applications for your project
- They can also be updated or extended for your own use case

Boosters

A booster is the implementation of a mission in a specific runtime. Boosters are preconfigured, functioning applications that demonstrate core principles of modern application development and run in an environment similar to production.

Each mission is implemented in one or more runtimes. Both the specific implementation and the actual project that contains your code are called a booster.

For example, the REST API Level 0 mission is implemented for these runtimes:

- Node.js booster
- Spring Boot booster
- Eclipse Vert.x booster
- Thorntail booster

CHAPTER 3. AVAILABLE MISSIONS AND BOOSTERS FOR THORNTAIL

The following boosters are available for Thorntail.

3.1. REST API LEVEL 0 MISSION - THORNTAIL BOOSTER

Mission proficiency level: **Foundational**.

What the REST API Level 0 Mission Does

The REST API Level 0 mission shows how to map business operations to a remote procedure call endpoint over HTTP using a REST framework. This corresponds to Level 0 in the Richardson Maturity Model. Creating an HTTP endpoint using REST and its underlying principles to define your API lets you quickly prototype and design the API flexibly.

This booster introduces the mechanics of interacting with a remote service using the HTTP protocol. It allows you to:

- Execute an HTTP **`GET`** request on the **`api/greeting`** endpoint.
- Receive a response in JSON format with a payload consisting of the **`Hello, World!`** String.
- Execute an HTTP **`GET`** request on the **`api/greeting`** endpoint while passing in a String argument. This uses the **`name`** request parameter in the query string.
- Receive a response in JSON format with a payload of **`Hello, $name!`** with **`$name`** replaced by the value of the **`name`** parameter passed into the request.

3.1.1. Viewing the booster source code and README

Prerequisites

One of the following:

- Access to developers.redhat.com/launch
- Fabric8 Launcher installed on a Single-node OpenShift Cluster

Procedure

1. Use the Fabric8 Launcher tool to generate your own version of the booster.
2. View the generated GitHub repository or download and extract the ZIP file that contains the booster source code.

Additional resources

- Using developers.redhat.com/launch
- Using the Fabric8 Launcher tool on a Single-node OpenShift Cluster

3.1.2. REST API Level 0 design tradeoffs

Table 3.1. Design Tradeoffs

Pros	Cons

• The booster enables fast prototyping.
• The API Design is flexible.
• HTTP endpoints allow clients to be language-neutral.
	• As an application or service matures, the REST API Level 0 approach might not scale well. It might not support a clean API design or use cases with database interactions.
	  ◦ Any operations involving shared, mutable state must be integrated with an appropriate backing datastore.
	  ◦ All requests handled by this API design are scoped only to the container servicing the request. Subsequent requests might not be served by the same container.

3.1.3. Deploying the REST API Level 0 booster to OpenShift Online

Use one of the following options to execute the REST API Level 0 booster on OpenShift Online.

- Use developers.redhat.com/launch
- Use the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using developers.redhat.com/launch provides an automated booster deployment workflow that executes the **oc** commands for you.

3.1.3.1. Deploying the booster using developers.redhat.com/launch

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the developers.redhat.com/launch URL in a browser and log in.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.1.3.2. Authenticating the oc CLI client

To work with boosters on OpenShift Online using the **oc** command-line client, you need to authenticate the client using the token provided by the OpenShift Online web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the OpenShift Online URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.1.3.3. Deploying the REST API Level 0 booster using the `oc` CLI client

Prerequisites

- The booster application created using developers.redhat.com/launch. For more information, see Section 3.1.3.1, “Deploying the booster using developers.redhat.com/launch”.
- The **oc** client authenticated. For more information, see Section 3.1.3.2, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new project in OpenShift.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.
4. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

   This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

5. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                READY     STATUS        RESTARTS
AGE
```

```
MY_APP_NAME-1-aaaaa               1/1       Running     0
58s
MY_APP_NAME-s2i-1-build           0/1       Completed   0
2m
```

The **MY_APP_NAME-1-aaaaa** pod should have a status of **Running** once it is fully deployed and started. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                  HOST/PORT
PATH      SERVICES         PORT      TERMINATION
MY_APP_NAME          MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME      8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME** as the base URL to access the application.

3.1.4. Deploying the REST API Level 0 booster to Single-node OpenShift Cluster

Use one of the following options to execute the REST API Level 0 booster locally on Single-node OpenShift Cluster:

- Using Fabric8 Launcher
- Using the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using Fabric8 Launcher provides an automated booster deployment workflow that executes the **oc** commands for you.

3.1.4.1. Getting the Fabric8 Launcher tool URL and credentials

You need the Fabric8 Launcher tool URL and user credentials to create and deploy boosters on Single-node OpenShift Cluster. This information is provided when the Single-node OpenShift Cluster is started.

Prerequisites

- The Fabric8 Launcher tool installed, configured, and running. For more information, see the Install and Configure the Fabric8 Launcher Tool guide.

Procedure

1. Navigate to the console where you started Single-node OpenShift Cluster.
2. Check the console output for the URL and user credentials you can use to access the running Fabric8 Launcher:

Example Console Output from a Single-node OpenShift Cluster Startup

```
...
-- Removing temporary directory ... OK
-- Server Information ...
   OpenShift server started.
   The server is accessible via web console at:
       https://192.168.42.152:8443

   You are logged in as:
       User:     developer
       Password: developer

   To login as administrator:
       oc login -u system:admin
```

3.1.4.2. Deploying the booster using the Fabric8 Launcher tool

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.1.4.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Fabric8 Launcher URL in a browser.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.1.4.3. Authenticating the oc CLI client

To work with boosters on Single-node OpenShift Cluster using the **oc** command-line client, you need to authenticate the client using the token provided by the Single-node OpenShift Cluster web interface.

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.1.4.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Single-node OpenShift Cluster URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your Single-node OpenShift Cluster account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.1.4.4. Deploying the REST API Level 0 booster using the `oc` CLI client

Prerequisites

- The booster application created using Fabric8 Launcher tool on a Single-node OpenShift Cluster. For more information, see Section 3.1.4.2, “Deploying the booster using the Fabric8 Launcher tool”.
- Your Fabric8 Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 3.1.4.3, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

   ```
   $ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
   ```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

   ```
   $ unzip MY_PROJECT_NAME.zip
   ```

2. Create a new project in OpenShift.

   ```
   $ oc new-project MY_PROJECT_NAME
   ```

3. Navigate to the root directory of your booster.

4. Use Maven to start the deployment to OpenShift.

   ```
   $ mvn clean fabric8:deploy -Popenshift
   ```

   This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

5. Check the status of your booster and ensure your pod is running.

   ```
   $ oc get pods -w
   NAME                              READY     STATUS      RESTARTS
   AGE
   MY_APP_NAME-1-aaaaa               1/1       Running     0
   58s
   MY_APP_NAME-s2i-1-build           0/1       Completed   0
   2m
   ```

   The **`MY_APP_NAME-1-aaaaa`** pod should have a status of **`Running`** once it is fully deployed and started. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                  HOST/PORT
PATH      SERVICES         PORT      TERMINATION
MY_APP_NAME          MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME      8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the application.

3.1.5. Deploying the REST API Level 0 booster to OpenShift Container Platform

The process of creating and deploying boosters to OpenShift Container Platform is similar to OpenShift Online:

Prerequisites

- The booster created using developers.redhat.com/launch or the Fabric8 Launcher tool.

Procedure

- Follow the instructions in Section 3.1.3, “Deploying the REST API Level 0 booster to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

3.1.6. Interacting with the unmodified REST API Level 0 booster for Thorntail

The booster provides a default HTTP endpoint that accepts GET requests.

Prerequisites

- Your application running
- The **`curl`** binary or a web browser

Procedure

1. Use **`curl`** to execute a **`GET`** request against the booster. You can also use a browser to do this.

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/greeting
{"content":"Hello, World!"}
```

2. Use **`curl`** to execute a **`GET`** request with the **`name`** URL parameter against the booster. You can also use a browser to do this.

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/greeting?name=Sarah
{"content":"Hello, Sarah!"}
```

NOTE

From a browser, you can also use a form provided by the booster to perform these same interactions. The form is located at the root of the project **`http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`**.

3.1.7. Running the REST API Level 0 booster integration tests

This booster includes a self-contained set of integration tests. When run inside an OpenShift project, the tests:

- Deploy a test instance of the application to the project.
- Execute the individual tests on that instance.
- Remove all instances of the application from the project when the testing is done.

WARNING

Executing integration tests removes all existing instances of the booster application from the target OpenShift project. To avoid accidentally removing your booster application, ensure that you create and select a separate OpenShift project to execute the tests.

Prerequisites

- The **oc** client authenticated
- An empty OpenShift project

Procedure

Execute the following command to run the integration tests:

```
$ mvn clean verify -Popenshift,openshift-it
```

3.1.8. REST resources

More background and related information on REST can be found here:

- Architectural Styles and the Design of Network-based Software Architectures - Representational State Transfer (REST)
- Richardson Maturity Model
- JSR 311: JAX-RS: The JavaTM API for RESTful Web Services
- RESTEasy Documentation
- REST API Level 0 Mission - Spring Boot Booster

- REST API Level 0 Mission - Eclipse Vert.x Booster
- REST API Level 0 Mission - Node.js Booster

3.2. EXTERNALIZED CONFIGURATION MISSION - THORNTAIL BOOSTER

Mission proficiency level: **Foundational**.

The Externalized Configuration mission provides a basic example of using a ConfigMap to externalize configuration. *ConfigMap* is an object used by OpenShift to inject configuration data as simple key and value pairs into one or more Linux containers while keeping the containers independent of OpenShift.

This mission shows you how to:

- Set up and configure a **`ConfigMap`**.
- Use the configuration provided by the **`ConfigMap`** within an application.
- Deploy changes to the **`ConfigMap`** configuration of running applications.

3.2.1. The externalized configuration design pattern

Whenever possible, externalize the application configuration and separate it from the application code. This allows the application configuration to change as it moves through different environments, but leaves the code unchanged. Externalizing the configuration also keeps sensitive or internal information out of your code base and version control. Many languages and application servers provide environment variables to support externalizing an application's configuration.

Microservices architectures and multi-language (polyglot) environments add a layer of complexity to managing an application's configuration. Applications consist of independent, distributed services, and each can have its own configuration. Keeping all configuration data synchronized and accessible creates a maintenance challenge.

ConfigMaps enable the application configuration to be externalized and used in individual Linux containers and pods on OpenShift. You can create a ConfigMap object in a variety of ways, including using a YAML file, and inject it into the Linux container. ConfigMaps also allow you to group and scale sets of configuration data. This lets you configure a large number of environments beyond the basic *Development*, *Stage*, and *Production*. You can find more information about ConfigMaps in the OpenShift documentation.

3.2.2. Externalized Configuration design tradeoffs

Table 3.2. Design Tradeoffs

Pros	Cons

Pros	Cons
• Configuration is separate from deployments • Can be updated independently • Can be shared across services	• Adding configuration to environment requires additional step • Has to be maintained separately • Requires coordination beyond the scope of a service

3.2.3. Viewing the booster source code and README

Prerequisites

One of the following:

- Access to developers.redhat.com/launch
- Fabric8 Launcher installed on a Single-node OpenShift Cluster

Procedure

1. Use the Fabric8 Launcher tool to generate your own version of the booster.
2. View the generated GitHub repository or download and extract the ZIP file that contains the booster source code.

Additional resources

- Using developers.redhat.com/launch
- Using the Fabric8 Launcher tool on a Single-node OpenShift Cluster

3.2.4. Deploying the Externalized Configuration booster to OpenShift Online

Use one of the following options to execute the Externalized Configuration booster on OpenShift Online.

- Use developers.redhat.com/launch
- Use the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using developers.redhat.com/launch provides an automated booster deployment workflow that executes the **oc** commands for you.

3.2.4.1. Deploying the booster using developers.redhat.com/launch

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the developers.redhat.com/launch URL in a browser and log in.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.2.4.2. Authenticating the `oc` CLI client

To work with boosters on OpenShift Online using the **oc** command-line client, you need to authenticate the client using the token provided by the OpenShift Online web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the OpenShift Online URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.2.4.3. Deploying the Externalized Configuration booster using the `oc` CLI client

Prerequisites

- The booster application created using developers.redhat.com/launch. For more information, see Section 3.2.4.1, “Deploying the booster using developers.redhat.com/launch”.
- The **oc** client authenticated. For more information, see Section 3.2.4.2, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.
4. Deploy your ConfigMap configuration to OpenShift using **`app-config.yml`** in the root of the booster.

```
$ oc create configmap app-config --from-file=app-config.yml
```

5. Verify your ConfigMap configuration has been deployed.

```
$ oc get configmap app-config -o yaml

apiVersion: v1
data:
  app-config.yml: |-
    greeting:
      message: Hello %s from a ConfigMap!
...
```

6. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift -DskipTests
```

This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

7. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                     READY     STATUS
RESTARTS   AGE
MY_APP_NAME-1-aaaaa               1/1       Running     0
58s
MY_APP_NAME-s2i-1-build           0/1       Completed   0
2m
```

The **`MY_APP_NAME-1-aaaaa`** pod should have a status of **`Running`** once its fully deployed and started. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

8. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                 HOST/PORT
PATH      SERVICES         PORT      TERMINATION
MY_APP_NAME         MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME      8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the application.

3.2.5. Deploying the Externalized Configuration booster to Single-node OpenShift Cluster

Use one of the following options to execute the Externalized Configuration booster locally on Single-node OpenShift Cluster:

- Using Fabric8 Launcher
- Using the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using Fabric8 Launcher provides an automated booster deployment workflow that executes the **oc** commands for you.

3.2.5.1. Getting the Fabric8 Launcher tool URL and credentials

You need the Fabric8 Launcher tool URL and user credentials to create and deploy boosters on Single-node OpenShift Cluster. This information is provided when the Single-node OpenShift Cluster is started.

Prerequisites

- The Fabric8 Launcher tool installed, configured, and running. For more information, see the Install and Configure the Fabric8 Launcher Tool guide.

Procedure

1. Navigate to the console where you started Single-node OpenShift Cluster.
2. Check the console output for the URL and user credentials you can use to access the running Fabric8 Launcher:

   Example Console Output from a Single-node OpenShift Cluster Startup

```
...
-- Removing temporary directory ... OK
-- Server Information ...
   OpenShift server started.
   The server is accessible via web console at:
       https://192.168.42.152:8443

   You are logged in as:
       User:     developer
       Password: developer

   To login as administrator:
       oc login -u system:admin
```

3.2.5.2. Deploying the booster using the Fabric8 Launcher tool

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.2.5.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Fabric8 Launcher URL in a browser.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.2.5.3. Authenticating the `oc` CLI client

To work with boosters on Single-node OpenShift Cluster using the **oc** command-line client, you need to authenticate the client using the token provided by the Single-node OpenShift Cluster web interface.

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.2.5.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Single-node OpenShift Cluster URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your Single-node OpenShift Cluster account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.2.5.4. Deploying the Externalized Configuration booster using the `oc` CLI client

Prerequisites

- The booster application created using Fabric8 Launcher tool on a Single-node OpenShift Cluster. For more information, see Section 3.2.5.2, “Deploying the booster using the Fabric8 Launcher tool”.
- Your Fabric8 Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 3.2.5.3, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.

4. Deploy your ConfigMap configuration to OpenShift using **`app-config.yml`** in the root of the booster.

```
$ oc create configmap app-config --from-file=app-config.yml
```

5. Verify your ConfigMap configuration has been deployed.

```
$ oc get configmap app-config -o yaml

apiVersion: v1
data:
  app-config.yml: |-
    greeting:
      message: Hello %s from a ConfigMap!
...
```

6. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift -DskipTests
```

This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

7. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                     READY     STATUS
RESTARTS   AGE
MY_APP_NAME-1-aaaaa               1/1       Running     0
58s
MY_APP_NAME-s2i-1-build           0/1       Completed   0
2m
```

The **`MY_APP_NAME-1-aaaaa`** pod should have a status of **`Running`** once its fully deployed and started. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

8. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                 HOST/PORT
PATH      SERVICES         PORT      TERMINATION
MY_APP_NAME         MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME      8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the application.

3.2.6. Deploying the Externalized Configuration booster to OpenShift Container Platform

The process of creating and deploying boosters to OpenShift Container Platform is similar to OpenShift Online:

Prerequisites

- The booster created using developers.redhat.com/launch or the Fabric8 Launcher tool.

Procedure

- Follow the instructions in Section 3.2.4, “Deploying the Externalized Configuration booster to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

3.2.7. Interacting with the unmodified Externalized Configuration booster for Thorntail

The booster provides a default HTTP endpoint that accepts GET requests.

Prerequisites

- Your application running
- The **`curl`** binary or a web browser

Procedure

1. Use **`curl`** to execute a **`GET`** request against the booster. You can also use a browser to do this.

```
$ curl http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/greeting
{"content":"Hello World from a ConfigMap!"}
```

2. Update the deployed ConfigMap configuration.

```
$ oc edit configmap app-config
```

Change the value for the **`greeting.message`** key to **`Bonjour %s from a ConfigMap!`** and save the file. After you save this, the changes will be propagated to your OpenShift instance.

3. Rollout the new version of your application so the ConfigMap configuration changes are picked up.

```
$ oc rollout latest dc/MY_APP_NAME
```

4. Check the status of your booster and ensure your new pod is running.

```
$ oc get pods -w
NAME                                READY     STATUS      RESTARTS
AGE
MY_APP_NAME-1-aaaaa       1/1       Running     0          58s
MY_APP_NAME-s2i-1-build   0/1       Completed   0          2m
```

The **MY_APP_NAME-1-aaaaa** pod should have a status of **Running** once its fully deployed and started. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

5. Execute a **GET** request using **curl** against the booster with the updated ConfigMap configuration to see your updated greeting. You can also do this from your browser using the web form provided by the application.

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/greeting
{"content":"Bonjour World from a ConfigMap!"}
```

3.2.8. Running the Externalized Configuration booster integration tests

This booster includes a self-contained set of integration tests. When run inside an OpenShift project, the tests:

- Deploy a test instance of the application to the project.
- Execute the individual tests on that instance.
- Remove all instances of the application from the project when the testing is done.

WARNING

Executing integration tests removes all existing instances of the booster application from the target OpenShift project. To avoid accidentally removing your booster application, ensure that you create and select a separate OpenShift project to execute the tests.

Prerequisites

- The **oc** client authenticated
- An empty OpenShift project

Procedure

Execute the following command to run the integration tests:

```
$ mvn clean verify -Popenshift,openshift-it
```

3.2.9. Externalized Configuration resources

More background and related information on Externalized Configuration and ConfigMap can be found here:

- OpenShift ConfigMap Documentation
- Blog Post about ConfigMap in OpenShift
- Externalized Configuration with Thorntail
- Externalized Configuration - Spring Boot Booster
- Externalized Configuration - Eclipse Vert.x Booster
- Externalized Configuration - Node.js Booster

3.3. RELATIONAL DATABASE BACKEND MISSION - THORNTAIL BOOSTER

Limitation: Run this booster on a Single-node OpenShift Cluster. You can also use a manual workflow to deploy this booster to OpenShift Online Pro and OpenShift Container Platform. This booster is not currently available on OpenShift Online Starter.

Mission proficiency level: **Foundational**.

What the Relational Database Backend Booster Does

The Relational Database Backend booster expands on the REST API Level 0 booster to provide a basic example of performing *create*, *read*, *update* and *delete* (*CRUD*) operations on a PostgreSQL database using a simple HTTP API. *CRUD* operations are the four basic functions of persistent storage, widely used when developing an HTTP API dealing with a database.

The booster also demonstrates the ability of the HTTP application to locate and connect to a database in OpenShift. Each runtime shows how to implement the connectivity solution best suited in the given case. The runtime can choose between options such as using *JDBC*, *JPA*, or accessing *ORM* APIs directly.

The booster application exposes an HTTP API, which provides endpoints that allow you to manipulate data by performing *CRUD* operations over HTTP. The *CRUD* operations are mapped to HTTP **`Verbs`**. The API uses JSON formatting to receive requests and return responses to the user. The user can also use an UI provided by the booster to use the application. Specifically, this booster provides an application that allows you to:

- Navigate to the application web interface in your browser. This exposes a simple website allowing you to perform *CRUD* operations on the data in the **`my_data`** database.
- Execute an HTTP **`GET`** request on the **`api/fruits`** endpoint.
- Receive a response formatted as a JSON array containing the list of all fruits in the database.
- Execute an HTTP **`GET`** request on the **`api/fruits/*`** endpoint while passing in a valid item ID as an argument.
- Receive a response in JSON format containing the name of the fruit with the given ID. If no item matches the specified ID, the call results in an HTTP error 404.
- Execute an HTTP **`POST`** request on the **`api/fruits`** endpoint passing in a valid **`name`** value to create a new entry in the database.

- Execute an HTTP **`PUT`** request on the **`api/fruits/*`** endpoint passing in a valid ID and a name as an argument. This updates the name of the item with the given ID to match the name specified in your request.
- Execute an HTTP **`DELETE`** request on the **`api/fruits/*`** endpoint, passing in a valid ID as an argument. This removes the item with the specified ID from the database and returns an HTTP code **`204`** (No Content) as a response. If you pass in an invalid ID, the call results in an HTTP error **`404`**.

This booster also contains a set of automated integration tests that can be used to verify that the application is fully integrated with the database.

This booster does not showcase a fully matured RESTful model (level 3), but it does use compatible HTTP verbs and status, following the recommended HTTP API practices.

3.3.1. Relational Database Backend design tradeoffs

Table 3.3. Design Tradeoffs

Pros	Cons
• Each runtime determines how to implement the database interactions. One can use a low-level connectivity API such as JDBC, some other can use JPA, and yet another can access ORM APIs directly. Each runtime decides what would be the best way. • Each runtime determines how the schema is created.	• The PostgreSQL database example provided with this mission is not backed up with persistent storage. Changes to the database are lost if you stop or redeploy the database pod. To use an external database with your mission's pod in order to preserve changes, see the Integrating External Services chapter of the OpenShift Documentation. It is also possible to set up persistent storage with database containers on OpenShift. (For more details about using persistent storage with OpenShift and containers, see the Persistent Storage, Managing Volumes and Persistent Volumes chapters of the OpenShift Documentation).

3.3.2. Viewing the booster source code and README

Prerequisites

One of the following:

- Access to developers.redhat.com/launch
- Fabric8 Launcher installed on a Single-node OpenShift Cluster

Procedure

1. Use the Fabric8 Launcher tool to generate your own version of the booster.
2. View the generated GitHub repository or download and extract the ZIP file that contains the booster source code.

Additional resources

- Using developers.redhat.com/launch
- Using the Fabric8 Launcher tool on a Single-node OpenShift Cluster

3.3.3. Deploying the Relational Database Backend booster to OpenShift Online

Use one of the following options to execute the Relational Database Backend booster on OpenShift Online.

- Use developers.redhat.com/launch
- Use the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using developers.redhat.com/launch provides an automated booster deployment workflow that executes the **oc** commands for you.

3.3.3.1. Deploying the booster using developers.redhat.com/launch

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the developers.redhat.com/launch URL in a browser and log in.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.3.3.2. Authenticating the `oc` CLI client

To work with boosters on OpenShift Online using the **oc** command-line client, you need to authenticate the client using the token provided by the OpenShift Online web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the OpenShift Online URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.3.3.3. Deploying the Relational Database Backend booster using the `oc` CLI client

Prerequisites

- The booster application created using developers.redhat.com/launch. For more information, see Section 3.3.3.1, “Deploying the booster using developers.redhat.com/launch”.
- The **oc** client authenticated. For more information, see Section 3.3.3.2, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

   ```
   $ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
   ```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

   ```
   $ unzip MY_PROJECT_NAME.zip
   ```

2. Create a new OpenShift project.

   ```
   $ oc new-project MY_PROJECT_NAME
   ```

3. Navigate to the root directory of your booster.

4. Deploy the PostgreSQL database to OpenShift. Ensure that you use the following values for user name, password, and database name when creating your database application. The booster application is pre-configured to use these values. Using different values prevents your booster application from integrating with the database.

   ```
   $ oc new-app -e POSTGRESQL_USER=luke -ePOSTGRESQL_PASSWORD=secret -
   ePOSTGRESQL_DATABASE=my_data openshift/postgresql-92-centos7 --
   name=my-database
   ```

5. Check the status of your database and ensure the pod is running.

   ```
   $ oc get pods -w
   my-database-1-aaaaa   1/1       Running   0         45s
   my-database-1-deploy   0/1       Completed   0         53s
   ```

   The **`my-database-1-aaaaa`** pod should have a status of **`Running`** and should be indicated as ready once it is fully deployed and started. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Use maven to start the deployment to OpenShift.

   ```
   $ mvn clean fabric8:deploy -Popenshift
   ```

This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

7. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                READY     STATUS      RESTARTS
AGE
MY_APP_NAME-1-aaaaa       1/1       Running     0          58s
MY_APP_NAME-s2i-1-build   0/1       Completed   0          2m
```

Your **MY_APP_NAME-1-aaaaa** pod should have a status of **Running** and should be indicated as ready once it is fully deployed and started.

8. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                HOST/PORT
PATH      SERVICES             PORT      TERMINATION
MY_APP_NAME   MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME   8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME** as the base URL to access the application.

3.3.4. Deploying the Relational Database Backend booster to Single-node OpenShift Cluster

Use one of the following options to execute the Relational Database Backend booster locally on Single-node OpenShift Cluster:

- Using Fabric8 Launcher
- Using the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using Fabric8 Launcher provides an automated booster deployment workflow that executes the **oc** commands for you.

3.3.4.1. Getting the Fabric8 Launcher tool URL and credentials

You need the Fabric8 Launcher tool URL and user credentials to create and deploy boosters on Single-node OpenShift Cluster. This information is provided when the Single-node OpenShift Cluster is started.

Prerequisites

- The Fabric8 Launcher tool installed, configured, and running. For more information, see the Install and Configure the Fabric8 Launcher Tool guide.

Procedure

1. Navigate to the console where you started Single-node OpenShift Cluster.

2. Check the console output for the URL and user credentials you can use to access the running Fabric8 Launcher:

Example Console Output from a Single-node OpenShift Cluster Startup

```
...
-- Removing temporary directory ... OK
-- Server Information ...
   OpenShift server started.
   The server is accessible via web console at:
       https://192.168.42.152:8443

   You are logged in as:
       User:     developer
       Password: developer

   To login as administrator:
       oc login -u system:admin
```

3.3.4.2. Deploying the booster using the Fabric8 Launcher tool

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.3.4.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Fabric8 Launcher URL in a browser.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.3.4.3. Authenticating the `oc` CLI client

To work with boosters on Single-node OpenShift Cluster using the **oc** command-line client, you need to authenticate the client using the token provided by the Single-node OpenShift Cluster web interface.

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.3.4.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Single-node OpenShift Cluster URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.

4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.

5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your Single-node OpenShift Cluster account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.3.4.4. Deploying the Relational Database Backend booster using the oc CLI client

Prerequisites

- The booster application created using Fabric8 Launcher tool on a Single-node OpenShift Cluster. For more information, see Section 3.3.4.2, “Deploying the booster using the Fabric8 Launcher tool”.
- Your Fabric8 Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 3.3.4.3, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.

4. Deploy the PostgreSQL database to OpenShift. Ensure that you use the following values for user name, password, and database name when creating your database application. The booster application is pre-configured to use these values. Using different values prevents your booster application from integrating with the database.

```
$ oc new-app -e POSTGRESQL_USER=luke -ePOSTGRESQL_PASSWORD=secret -
ePOSTGRESQL_DATABASE=my_data openshift/postgresql-92-centos7 --
name=my-database
```

5. Check the status of your database and ensure the pod is running.

```
$ oc get pods -w
my-database-1-aaaaa   1/1       Running   0         45s
my-database-1-deploy   0/1       Completed   0         53s
```

   The **`my-database-1-aaaaa`** pod should have a status of **`Running`** and should be indicated as

ready once it is fully deployed and started. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Use maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

7. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                READY     STATUS      RESTARTS
AGE
MY_APP_NAME-1-aaaaa       1/1       Running     0          58s
MY_APP_NAME-s2i-1-build   0/1       Completed   0          2m
```

Your **`MY_APP_NAME-1-aaaaa`** pod should have a status of **`Running`** and should be indicated as ready once it is fully deployed and started.

8. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                 HOST/PORT
PATH      SERVICES             PORT      TERMINATION
MY_APP_NAME   MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME   8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the application.

3.3.5. Deploying the Relational Database Backend booster to OpenShift Container Platform

The process of creating and deploying boosters to OpenShift Container Platform is similar to OpenShift Online:

Prerequisites

- The booster created using developers.redhat.com/launch or the Fabric8 Launcher tool.

Procedure

- Follow the instructions in Section 3.3.3, “Deploying the Relational Database Backend booster to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

3.3.6. Interacting with the Relational Database Backend API

When you have finished creating your application booster, you can interact with it the following way:

Prerequisites

- Your application running
- The **`curl`** binary or a web browser

Procedure

1. Obtain the URL of your application by executing the following command:

```
$ oc get route MY_APP_NAME
```

```
NAME                  HOST/PORT
PATH      SERVICES             PORT      TERMINATION
MY_APP_NAME           MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME           8080
```

2. To access the web interface of the database application, navigate to the *application URL* in your browser:

```
http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
```

Alternatively, you can make requests directly on the **`api/fruits/*`** endpoint using **`curl`**:

List all entries in the database:

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/fruits
```

```
[ {
  "id" : 1,
  "name" : "Cherry",
}, {
  "id" : 2,
  "name" : "Apple",
}, {
  "id" : 3,
  "name" : "Banana",
} ]
```

Retrieve an entry with a specific ID

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/fruits/3
```

```
{
  "id" : 3,
  "name" : "Banana",
}
```

Create a new entry:

Create a new entry:

```
$ curl -H "Content-Type: application/json" -X POST -d
'{"name":"pear"}'  http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/fruits
```

```
{
  "id" : 4,
  "name" : "pear",
}
```

Update an Entry

```
$ curl -H "Content-Type: application/json" -X PUT -d
'{"name":"pineapple"}'  http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/fruits/1
```

```
{
  "id" : 1,
  "name" : "pineapple",
}
```

Delete an Entry:

```
$ curl -X DELETE http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/fruits/1
```

Troubleshooting

- If you receive an HTTP Error code **503** as a response after executing these commands, it means that the application is not ready yet.

3.3.7. Running the Relational Database Backend booster integration tests

This booster includes a self-contained set of integration tests. When run inside an OpenShift project, the tests:

- Deploy a test instance of the application to the project.
- Execute the individual tests on that instance.
- Remove all instances of the application from the project when the testing is done.

WARNING

Executing integration tests removes all existing instances of the booster application from the target OpenShift project. To avoid accidentally removing your booster application, ensure that you create and select a separate OpenShift project to execute the tests.

Prerequisites

- The **oc** client authenticated
- An empty OpenShift project

Procedure

Execute the following command to run the integration tests:

```
$ mvn clean verify -Popenshift,openshift-it
```

3.3.8. Relational database resources

More background and related information on running relational databases in OpenShift, CRUD, HTTP API and REST can be found here:

- HTTP Verbs
- Architectural Styles and the Design of Network-based Software Architectures - Representational State Transfer (REST)
- The never ending REST API design debase
- REST APIs must be Hypertext driven
- Richardson Maturity Model
- JSR 311: JAX-RS: The JavaTM API for RESTful Web Services
- RESTEasy Documentation
- Relational Database Backend Mission - Spring Boot Booster
- Relational Database Backend Mission - Eclipse Vert.x Booster
- Relational Database Backend Mission - Node.js Booster

3.4. HEALTH CHECK MISSION - THORNTAIL BOOSTER

Mission proficiency level: **Foundational**.

When you deploy an application, its important to know if it is available and if it can start handling incoming requests. Implementing the *health check* pattern allows you to monitor the health of an application, which includes if an application is available and whether it is able to service requests.

NOTE

If you are not familiar with the health check terminology, see the Section 3.4.1, “Health check concepts” section first.

The purpose of this use case is to demonstrate the health check pattern through the use of probing. Probing is used to report the liveness and readiness of an application. In this use case, you configure an application which exposes an HTTP **`health`** endpoint to issue HTTP requests. If the container is alive, according to the liveness probe on the **`health`** HTTP endpoint, the management platform receives **`200`**

as return code and no further action is required. If the **`health`** HTTP endpoint does not return a response, for example if the thread is blocked, then the application is not considered alive according to the liveness probe. In that case, the platform kills the pod corresponding to that application and recreates a new pod to restart the application.

This use case also allows you to demonstrate and use a readiness probe. In cases where the application is running but is unable to handle requests, such as when the application returns an HTTP **`503`** response code during restart, this application is not considered ready according to the readiness probe. If the application is not considered ready by the readiness probe, requests are not routed to that application until it is considered ready according to the readiness probe.

3.4.1. Health check concepts

In order to understand the health check pattern, you need to first understand the following concepts:

Liveness

Liveness defines whether an application is running or not. Sometimes a running application moves into an unresponsive or stopped state and needs to be restarted. Checking for liveness helps determine whether or not an application needs to be restarted.

Readiness

Readiness defines whether a running application can service requests. Sometimes a running application moves into an error or broken state where it can no longer service requests. Checking readiness helps determine whether or not requests should continue to be routed to that application.

Fail-over

Fail-over enables failures in servicing requests to be handled gracefully. If an application fails to service a request, that request and future requests can then *fail-over* or be routed to another application, which is usually a redundant copy of that same application.

Resilience and Stability

Resilience and Stability enable failures in servicing requests to be handled gracefully. If an application fails to service a request due to connection loss, in a resilient system that request can be retried after the connection is re-established.

Probe

A probe is a Kubernetes action that periodically performs diagnostics on a running container.

3.4.2. Viewing the booster source code and README

Prerequisites

One of the following:

- Access to developers.redhat.com/launch
- Fabric8 Launcher installed on a Single-node OpenShift Cluster

Procedure

1. Use the Fabric8 Launcher tool to generate your own version of the booster.
2. View the generated GitHub repository or download and extract the ZIP file that contains the booster source code.

Additional resources

- Using developers.redhat.com/launch
- Using the Fabric8 Launcher tool on a Single-node OpenShift Cluster

3.4.3. Deploying the Health Check booster to OpenShift Online

Use one of the following options to execute the Health Check booster on OpenShift Online.

- Use developers.redhat.com/launch
- Use the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using developers.redhat.com/launch provides an automated booster deployment workflow that executes the **oc** commands for you.

3.4.3.1. Deploying the booster using developers.redhat.com/launch

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the developers.redhat.com/launch URL in a browser and log in.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.4.3.2. Authenticating the oc CLI client

To work with boosters on OpenShift Online using the **oc** command-line client, you need to authenticate the client using the token provided by the OpenShift Online web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the OpenShift Online URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.4.3.3. Deploying the Health Check booster using the oc CLI client

Prerequisites

- The booster application created using developers.redhat.com/launch. For more information, see Section 3.4.3.1, “Deploying the booster using developers.redhat.com/launch”.
- The **oc** client authenticated. For more information, see Section 3.4.3.2, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.
4. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

   This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

5. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                READY     STATUS       RESTARTS
AGE
MY_APP_NAME-1-aaaaa                  1/1       Running     0
58s
MY_APP_NAME-s2i-1-build              0/1       Completed   0
2m
```

   The **MY_APP_NAME-1-aaaaa** pod should have a status of **Running** once its fully deployed and started. You should also wait for your pod to be ready before proceeding, which is shown in the **READY** column. For example, **MY_APP_NAME-1-aaaaa** is ready when the **READY** column is **1/1**. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Once your booster is deployed and started, determine its route.

   Example Route Information

```
$ oc get routes
```

```
NAME                 HOST/PORT
PATH      SERVICES         PORT       TERMINATION
MY_APP_NAME          MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME       8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the application.

3.4.4. Deploying the Health Check booster to Single-node OpenShift Cluster

Use one of the following options to execute the Health Check booster locally on Single-node OpenShift Cluster:

- Using Fabric8 Launcher
- Using the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using Fabric8 Launcher provides an automated booster deployment workflow that executes the **oc** commands for you.

3.4.4.1. Getting the Fabric8 Launcher tool URL and credentials

You need the Fabric8 Launcher tool URL and user credentials to create and deploy boosters on Single-node OpenShift Cluster. This information is provided when the Single-node OpenShift Cluster is started.

Prerequisites

- The Fabric8 Launcher tool installed, configured, and running. For more information, see the Install and Configure the Fabric8 Launcher Tool guide.

Procedure

1. Navigate to the console where you started Single-node OpenShift Cluster.
2. Check the console output for the URL and user credentials you can use to access the running Fabric8 Launcher:

Example Console Output from a Single-node OpenShift Cluster Startup

```
...
-- Removing temporary directory ... OK
-- Server Information ...
   OpenShift server started.
   The server is accessible via web console at:
       https://192.168.42.152:8443

   You are logged in as:
       User:     developer
       Password: developer

   To login as administrator:
       oc login -u system:admin
```

3.4.4.2. Deploying the booster using the Fabric8 Launcher tool

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.4.4.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Fabric8 Launcher URL in a browser.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.4.4.3. Authenticating the `oc` CLI client

To work with boosters on Single-node OpenShift Cluster using the **oc** command-line client, you need to authenticate the client using the token provided by the Single-node OpenShift Cluster web interface.

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.4.4.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Single-node OpenShift Cluster URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your Single-node OpenShift Cluster account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.4.4.4. Deploying the Health Check booster using the `oc` CLI client

Prerequisites

- The booster application created using Fabric8 Launcher tool on a Single-node OpenShift Cluster. For more information, see Section 3.4.4.2, “Deploying the booster using the Fabric8 Launcher tool”.
- Your Fabric8 Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 3.4.4.3, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.

4. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

   This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

5. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                READY     STATUS      RESTARTS
AGE
MY_APP_NAME-1-aaaaa                 1/1       Running     0
58s
MY_APP_NAME-s2i-1-build             0/1       Completed   0
2m
```

   The **MY_APP_NAME-1-aaaaa** pod should have a status of **Running** once its fully deployed and started. You should also wait for your pod to be ready before proceeding, which is shown in the **READY** column. For example, **MY_APP_NAME-1-aaaaa** is ready when the **READY** column is **1/1**. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                 HOST/PORT
PATH      SERVICES         PORT      TERMINATION
MY_APP_NAME          MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME      8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME** as the base URL to access the application.

3.4.5. Deploying the Health Check booster to OpenShift Container Platform

The process of creating and deploying boosters to OpenShift Container Platform is similar to OpenShift Online:

Prerequisites

- The booster created using developers.redhat.com/launch or the Fabric8 Launcher tool.

Procedure

- Follow the instructions in Section 3.4.3, “Deploying the Health Check booster to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

3.4.6. Interacting with the unmodified Health Check booster

Once you have the booster deployed, you will have a service called **`MY_APP_NAME`** running that exposes the following REST endpoints:

/api/greeting
: Returns a name as a String.

/api/stop
: Forces the service to become unresponsive as means to simulate a failure.

The following steps demonstrate how to verify the service availability and simulate a failure. This failure of an available service causes the OpenShift self-healing capabilities to be trigger on the service.

Alternatively, you can use the web interface to perform these steps.

1. Use **`curl`** to execute a **`GET`** request against the **`MY_APP_NAME`** service. You can also use a browser to do this.

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/greeting
```

```
{"content":"Hello, World!"}
```

2. Invoke the **`/api/stop`** endpoint and verify the availability of the **`/api/greeting`** endpoint shortly after that.
   Invoking the **`/api/stop`** endpoint simulates an internal service failure and triggers the OpenShift self-healing capabilities. When invoking **`/api/greeting`** after simulating the failure, the service should return a HTTP status **`503`**.

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/stop
```

   (followed by)

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/greeting
```

```
<html>
  <head><title>Error</title></head>
  <body>503 - Service Unavailable</body>
</html>
```

3. Use **`oc get pods -w`** to continuously watch the self-healing capabilities in action.
   While invoking the service failure, you can watch the self-healing capabilities in action on OpenShift console, or with the **`oc`** client tools. You should see the number of pods in the **READY** state move to zero (**`0/1`**) and after a short period (less than one minute) move back up to one (**`1/1`**). In addition to that, the **RESTARTS** count increases every time you you invoke the service failure.

```
$ oc get pods -w
NAME                              READY     STATUS    RESTARTS   AGE
MY_APP_NAME-1-26iy7   0/1       Running   5          18m
MY_APP_NAME-1-26iy7   1/1       Running   5         19m
```

4. Optional: Use the web interface to invoke the service.
   Alternatively to the interaction using the terminal window, you can use the web interface provided by the service to invoke the different methods and watch the service move through the life cycle phases.

```
http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
```

5. Optional: Use the web console to view the log output generated by the application at each stage of the self-healing process.

   1. Navigate to your project.
   2. On the sidebar, click on *Monitoring*.
   3. In the upper right-hand corner of the screen, click on *Events* to display the log messages.
   4. Optional: Click *View Details* to display a detailed view of the Event log.

   The health check application generates the following messages:

Message	Status
Unhealthy	Readiness probe failed. This message is expected and indicates that the simulated failure of the **`/api/greeting`** endpoint has been detected and the self-healing process starts.
Killing	The unavailable Docker container running the service is being killed before being re-created.
Pulling	Downloading the latest version of docker image to re-create the container.
Pulled	Docker image downloaded successfully.

Message	Status
Created	Docker container has been successfully created
Started	Docker container is ready to handle requests

3.4.7. Running the Health Check booster integration tests

This booster includes a self-contained set of integration tests. When run inside an OpenShift project, the tests:

- Deploy a test instance of the application to the project.
- Execute the individual tests on that instance.
- Remove all instances of the application from the project when the testing is done.

WARNING

Executing integration tests removes all existing instances of the booster application from the target OpenShift project. To avoid accidentally removing your booster application, ensure that you create and select a separate OpenShift project to execute the tests.

Prerequisites

- The **oc** client authenticated
- An empty OpenShift project

Procedure

Execute the following command to run the integration tests:

```
$ mvn clean verify -Popenshift,openshift-it
```

3.4.8. Health check resources

More background and related information on health checking can be found here:

- Overview of Application Health in OpenShift
- Health Checking in Kubernetes
- Kubernetes Liveness and Readiness Probes
- Kubernetes Probes
- Health Check - Spring Boot Booster

- Health Check - Eclipse Vert.x Booster
- Health Check - Node.js Booster

3.5. CIRCUIT BREAKER MISSION - THORNTAIL BOOSTER

Limitation: Run this booster on a Single-node OpenShift Cluster. You can also use a manual workflow to deploy this booster to OpenShift Online Pro and OpenShift Container Platform. This booster is not currently available on OpenShift Online Starter.

Mission proficiency level: **Foundational**.

The *Circuit Breaker* mission demonstrates a generic pattern for reporting the failure of a service and then limiting access to the failed service until it becomes available to handle requests. This helps prevent cascading failure in other services that depend on the failed services for functionality.

This mission shows you how to implement a Circuit Breaker and Fallback pattern in your services.

3.5.1. The circuit breaker design pattern

The Circuit Breaker is a pattern intended to:

- Reduce the impact of network failure and high latency on service architectures where services synchronously invoke other services.
  If one of the services:
  - becomes unavailable due to network failure, or
  - incurs unusually high latency values due to overwhelming traffic,

  other services attempting to call its endpoint may end up exhausting critical resources in an attempt to reach it, rendering themselves unusable.
- Prevent the condition also known as cascading failure, which can render the entire microservice architecture unusable.
- Act as a proxy between a protected function and a remote function, which monitors for failures.
- Trip once the failures reach a certain threshold, and all further calls to the circuit breaker return an error or a predefined fallback response, without the protected call being made at all.

The Circuit Breaker usually also contain an error reporting mechanism that notifies you when the Circuit Breaker trips.

Circuit breaker implementation

- With the Circuit Breaker pattern implemented, a service client invokes a remote service endpoint via a proxy at regular intervals.
- If the calls to the remote service endpoint fail repeatedly and consistently, the Circuit Breaker trips, making all calls to the service fail immediately over a set timeout period and returns a predefined fallback response.
- When the timeout period expires, a limited number of test calls are allowed to pass through to the remote service to determine whether it has healed, or remains unavailable.

- If the test calls fail, the Circuit Breaker keeps the service unavailable and keeps returning the fallback responses to incoming calls.
- If the test calls succeed, the Circuit Breaker closes, fully enabling traffic to reach the remote service again.

3.5.2. Circuit Breaker design tradeoffs

Table 3.4. Design Tradeoffs

Pros	Cons

• Enables a service to handle the failure of other services it invokes.	• Optimizing the timeout values can be challenging
	  ◦ Larger-than-necessary timeout values may generate excessive latency.
	  ◦ Smaller-than-necessary timeout values may introduce false positives.

3.5.3. Viewing the booster source code and README

Prerequisites

One of the following:

- Access to developers.redhat.com/launch
- Fabric8 Launcher installed on a Single-node OpenShift Cluster

Procedure

1. Use the Fabric8 Launcher tool to generate your own version of the booster.
2. View the generated GitHub repository or download and extract the ZIP file that contains the booster source code.

Additional resources

- Using developers.redhat.com/launch
- Using the Fabric8 Launcher tool on a Single-node OpenShift Cluster

3.5.4. Deploying the Circuit Breaker booster to OpenShift Online

Use one of the following options to execute the Circuit Breaker booster on OpenShift Online.

- Use developers.redhat.com/launch
- Use the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using developers.redhat.com/launch provides an automated booster deployment workflow that executes the **oc** commands for you.

3.5.4.1. Deploying the booster using developers.redhat.com/launch

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the developers.redhat.com/launch URL in a browser and log in.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.5.4.2. Authenticating the `oc` CLI client

To work with boosters on OpenShift Online using the **oc** command-line client, you need to authenticate the client using the token provided by the OpenShift Online web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the OpenShift Online URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.5.4.3. Deploying the Circuit Breaker booster using the `oc` CLI client

Prerequisites

- The booster application created using developers.redhat.com/launch. For more information, see Section 3.5.4.1, “Deploying the booster using developers.redhat.com/launch”.
- The **oc** client authenticated. For more information, see Section 3.5.4.2, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.

4. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

   This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

5. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                              READY     STATUS      RESTARTS
AGE
MY_APP_NAME-greeting-1-aaaaa      1/1       Running   0
17s
MY_APP_NAME-greeting-1-deploy     0/1       Completed 0
22s
MY_APP_NAME-name-1-aaaaa          1/1       Running   0
14s
MY_APP_NAME-name-1-deploy         0/1       Completed 0
28s
```

   Both the **MY_APP_NAME-greeting-1-aaaaa** and **MY_APP_NAME-name-1-aaaaa** pods should have a status of **Running** once they are fully deployed and started. You should also wait for your pods to be ready before proceeding, which is shown in the **READY** column. For example, **MY_APP_NAME-greeting-1-aaaaa** is ready when the **READY** column is **1/1**. Your specific pod names will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                  HOST/PORT
PATH      SERVICES        PORT      TERMINATION
MY_APP_NAME-greeting   MY_APP_NAME-greeting-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME            MY_APP_NAME-greeting
8080                    None
```

```
MY_APP_NAME-name       MY_APP_NAME-name-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME            MY_APP_NAME-name
8080                    None
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-greeting-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the application.

3.5.5. Deploying the Circuit Breaker booster to Single-node OpenShift Cluster

Use one of the following options to execute the Circuit Breaker booster locally on Single-node OpenShift Cluster:

- Using Fabric8 Launcher
- Using the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using Fabric8 Launcher provides an automated booster deployment workflow that executes the **oc** commands for you.

3.5.5.1. Getting the Fabric8 Launcher tool URL and credentials

You need the Fabric8 Launcher tool URL and user credentials to create and deploy boosters on Single-node OpenShift Cluster. This information is provided when the Single-node OpenShift Cluster is started.

Prerequisites

- The Fabric8 Launcher tool installed, configured, and running. For more information, see the Install and Configure the Fabric8 Launcher Tool guide.

Procedure

1. Navigate to the console where you started Single-node OpenShift Cluster.
2. Check the console output for the URL and user credentials you can use to access the running Fabric8 Launcher:

Example Console Output from a Single-node OpenShift Cluster Startup

```
...
-- Removing temporary directory ... OK
-- Server Information ...
   OpenShift server started.
   The server is accessible via web console at:
       https://192.168.42.152:8443

   You are logged in as:
       User:     developer
       Password: developer

   To login as administrator:
       oc login -u system:admin
```

3.5.5.2. Deploying the booster using the Fabric8 Launcher tool

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.5.5.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Fabric8 Launcher URL in a browser.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.5.5.3. Authenticating the `oc` CLI client

To work with boosters on Single-node OpenShift Cluster using the **oc** command-line client, you need to authenticate the client using the token provided by the Single-node OpenShift Cluster web interface.

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.5.5.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Single-node OpenShift Cluster URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your Single-node OpenShift Cluster account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.5.5.4. Deploying the Circuit Breaker booster using the `oc` CLI client

Prerequisites

- The booster application created using Fabric8 Launcher tool on a Single-node OpenShift Cluster. For more information, see Section 3.5.5.2, “Deploying the booster using the Fabric8 Launcher tool”.
- Your Fabric8 Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 3.5.5.3, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.

4. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

   This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

5. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                              READY     STATUS      RESTARTS
AGE
MY_APP_NAME-greeting-1-aaaaa      1/1       Running   0
17s
MY_APP_NAME-greeting-1-deploy     0/1       Completed 0
22s
MY_APP_NAME-name-1-aaaaa          1/1       Running   0
14s
MY_APP_NAME-name-1-deploy         0/1       Completed 0
28s
```

   Both the **`MY_APP_NAME-greeting-1-aaaaa`** and **`MY_APP_NAME-name-1-aaaaa`** pods should have a status of **`Running`** once they are fully deployed and started. You should also wait for your pods to be ready before proceeding, which is shown in the **`READY`** column. For example, **`MY_APP_NAME-greeting-1-aaaaa`** is ready when the **`READY`** column is **`1/1`**. Your specific pod names will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Once your booster is deployed and started, determine its route.

   Example Route Information

```
$ oc get routes
NAME                  HOST/PORT
PATH      SERVICES        PORT      TERMINATION
MY_APP_NAME-greeting   MY_APP_NAME-greeting-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME            MY_APP_NAME-greeting
8080                    None
```

```
MY_APP_NAME-name        MY_APP_NAME-name-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME              MY_APP_NAME-name
8080                     None
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-greeting-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the application.

3.5.6. Deploying the Circuit Breaker booster to OpenShift Container Platform

The process of creating and deploying boosters to OpenShift Container Platform is similar to OpenShift Online:

Prerequisites

- The booster created using developers.redhat.com/launch or the Fabric8 Launcher tool.

Procedure

- Follow the instructions in Section 3.5.4, “Deploying the Circuit Breaker booster to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

3.5.7. Interacting with the unmodified Thorntail Circuit Breaker booster

Once you have the Thorntail booster deployed, you have the following services running:

`MY_APP_NAME-name`

Exposes the following endpoints:

- the **`/api/name`** endpoint, which returns a name when this service is working, and an error when this service is set up to demonstrate failure.
- the **`/api/state`** endpoint, which controls the behavior of the **`/api/name`** endpoint and determines whether the service works correctly or demonstrates failure.

`MY_APP_NAME-greeting`

Exposes the following endpoints:

- the **`/api/greeting`** endpoint that you can call to get a personalized greeting response. When you call the **`/api/greeting`** endpoint, it issues a call against the **`/api/name`** endpoint of the **`MY_APP_NAME-name`** service as part of processing your request. The call made against the **`/api/name`** endpoint is protected by the Circuit Breaker.

  If the remote endpoint is available, the **`name`** service responds with an HTTP code **`200`** (**`OK`**) and you receive the following greeting from the **`/api/greeting`** endpoint:

  ```
  {"content":"Hello, World!"}
  ```

  If the remote endpoint is unavailable, the **`name`** service responds with an HTTP code **`500`** (**`Internal server error`**) and you receive a predefined fallback response from the **`/api/greeting`** endpoint:

```
{"content":"Hello, Fallback!"}
```

- the **/api/cb-state** endpoint, which returns the state of the Circuit Breaker. The state can be:
  - *open* : the circuit breaker is preventing requests from reaching the failed service,
  - *closed*: the circuit breaker is allowing requests to reach the service.

The following steps demonstrate how to verify the availability of the service, simulate a failure and receive a fallback response.

1. Use **curl** to execute a **GET** request against the **MY_APP_NAME-greeting** service. You can also use the **Invoke** button in the web interface to do this.

```
$ curl http://MY_APP_NAME-greeting-
MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME/api/greeting
{"content":"Hello, World!"}
```

2. To simulate the failure of the **MY_APP_NAME-name** service you can:
   - use the **Toggle** button in the web interface.
   - scale the number of replicas of the pod running the **MY_APP_NAME-name** service down to 0.
   - execute an HTTP **PUT** request against the **/api/state** endpoint of the **MY_APP_NAME-name** service to set its state to **fail**.

```
$ curl -X PUT -H "Content-Type: application/json" -d '{"state":
"fail"}' http://MY_APP_NAME-name-
MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME/api/state
```

3. Invoke the **/api/greeting** endpoint. When several requests on the **/api/name** endpoint fail:
   a. the Circuit Breaker opens,
   b. the state indicator in the web interface changes from **CLOSED** to **OPEN**,
   c. the Circuit Breaker issues a fallback response when you invoke the **/api/greeting** endpoint:

```
$ curl http://MY_APP_NAME-greeting-
MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME/api/greeting
{"content":"Hello, Fallback!"}
```

4. Restore the name **MY_APP_NAME-name** service to availability. To do this you can:
   - use the **Toggle** button in the web interface.
   - scale the number of replicas of the pod running the **MY_APP_NAME-name** service back up to 1.
   - execute an HTTP **PUT** request against the **/api/state** endpoint of the **MY_APP_NAME-name** service to set its state back to **ok**.

```
$ curl -X PUT -H "Content-Type: application/json" -d '{"state":
"ok"}' http://MY_APP_NAME-name-
MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME/api/state
```

5. Invoke the **`/api/greeting`** endpoint again. When several requests on the **`/api/name`** endpoint succeed:
   a. the Circuit Breaker closes,
   b. the state indicator in the web interface changes from **`OPEN`** to **`CLOSED`**,
   c. the Circuit Breaker issues a returns the **`Hello World!`** greeting when you invoke the **`/api/greeting`** endpoint:

```
$ curl http://MY_APP_NAME-greeting-
MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME/api/greeting
{"content":"Hello, World!"}
```

3.5.8. Running the Circuit Breaker booster integration tests

This booster includes a self-contained set of integration tests. When run inside an OpenShift project, the tests:

- Deploy a test instance of the application to the project.
- Execute the individual tests on that instance.
- Remove all instances of the application from the project when the testing is done.

WARNING

Executing integration tests removes all existing instances of the booster application from the target OpenShift project. To avoid accidentally removing your booster application, ensure that you create and select a separate OpenShift project to execute the tests.

Prerequisites

- The **oc** client authenticated
- An empty OpenShift project

Procedure

Execute the following command to run the integration tests:

```
$ mvn clean verify -Popenshift,openshift-it
```

3.5.9. Using Hystrix Dashboard to monitor the circuit breaker

Hystrix Dashboard lets you easily monitor the health of your services in real time by aggregating Hystrix metrics data from an event stream and displaying them on one screen.

Prerequisites

- The application deployed

Procedure

1. Log in to your Single-node OpenShift Cluster cluster.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

2. To access the Web console, use your browser to navigate to your Single-node OpenShift Cluster URL.

3. Navigate to the project that contains your Circuit Breaker application.

```
$ oc project MY_PROJECT_NAME
```

4. Import the YAML template for the Hystrix Dashboard application. You can do this by clicking *Add to Project*, then selecting the *Import YAML / JSON* tab, and copying the contents of the YAML file into the text box. Alternatively, you can execute the following command:

```
$ oc create -f https://raw.githubusercontent.com/snowdrop/openshift-
templates/master/hystrix-dashboard/hystrix-dashboard.yml
```

5. Click the *Create* button to create the Hystrix Dashboard application based on the template. Alternatively, you can execute the following command.

```
$ oc new-app --template=hystrix-dashboard
```

6. Wait for the pod containing Hystrix Dashboard to deploy.

7. Obtain the route of your Hystrix Dashboard application.

```
$ oc get route hystrix-dashboard
NAME                HOST/PORT
PATH      SERVICES            PORT      TERMINATION   WILDCARD
hystrix-dashboard   hystrix-dashboard-
MY_PROJECT_NAME.LOCAL_OPENSHIFT_HOSTNAME                hystrix-
dashboard   <all>                   None
```

8. To access the Dashboard, open the Dashboard application route URL in your browser. Alternatively, you can navigate to the *Overview* screen in the Web console and click the route URL in the header above the pod containing your Hystrix Dashboard application.

9. To use the Dashboard to monitor the **`MY_APP_NAME-greeting`** service, replace the default event stream address with the following address and click the *Monitor Stream* button.

```
http://MY_APP_NAME-greeting/hystrix.stream
```

Additional resources

- The Hystrix Dashboard wiki page

3.5.10. Circuit breaker resources

Follow the links below for more background information on the design principles behind the Circuit Breaker pattern

- microservices.io: Microservice Patterns: Circuit Breaker
- Martin Fowler: CircuitBreaker
- Circuit Breaker Mission - Spring Boot Booster
- Circuit Breaker Mission - Eclipse Vert.x Booster
- Circuit Breaker Mission - Node.js Booster

3.6. SECURED MISSION - THORNTAIL BOOSTER

Limitation: Run this booster on a Single-node OpenShift Cluster. You can also use a manual workflow to deploy this booster to OpenShift Online Pro and OpenShift Container Platform. This booster is not currently available on OpenShift Online Starter.

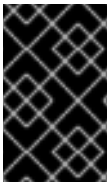
Mission proficiency level: **Advanced**.

The Secured booster secures a REST endpoint using Red Hat SSO. (This booster expands on the REST API Level 0 booster).

Red Hat SSO:

- Implements the Open ID Connect protocol which is an extension of the OAuth 2.0 specification.
- Issues access tokens to provide clients with various access rights to secured resources.

Securing an application with SSO enables you to add security to your applications while centralizing the security configuration.



IMPORTANT

This mission comes with Red Hat SSO pre-configured for demonstration purposes, it does not explain its principles, usage, or configuration. Before using this mission, ensure that you are familiar with the basic concepts related to Red Hat SSO.

3.6.1. The Secured project structure

The SSO booster project contains:

- the sources for the Greeting service, which is the one which we are going to to secure
- a template file (`service.sso.yaml`) to deploy the SSO server
- the Keycloak adapter configuration to secure the service

3.6.2. Viewing the booster source code and README

Prerequisites

One of the following:

- Access to developers.redhat.com/launch
- Fabric8 Launcher installed on a Single-node OpenShift Cluster

Procedure

1. Use the Fabric8 Launcher tool to generate your own version of the booster.
2. View the generated GitHub repository or download and extract the ZIP file that contains the booster source code.

Additional resources

- Using developers.redhat.com/launch
- Using the Fabric8 Launcher tool on a Single-node OpenShift Cluster

3.6.3. Red Hat SSO deployment configuration

The `service.sso.yaml` file in this booster contains all OpenShift configuration items to deploy a pre-configured Red Hat SSO server. The SSO server configuration has been simplified for the sake of this exercise and does provide an out-of-the-box configuration, with pre-configured users and security settings. The `service.sso.yaml` file also contains very long lines, and some text editors, such as gedit, may have issues reading this file.

WARNING

It is not recommended to use this SSO configuration in production. Specifically, the simplifications made to the booster security configuration impact the ability to use it in a production environment.

Table 3.5. SSO Booster Simplifications

Change	Reason	Recommendation
The default configuration includes both public and **private keys in the yaml configuration files**.	We did this because the end user can deploy Red Hat SSO module and have it in a usable state without needing to know the internals or how to configure Red Hat SSO.	In production, do not store private keys under source control. They should be added by the server administrator.

Change	Reason	Recommendation
The configured **clients accept any callback url**.	To avoid having a custom configuration for each runtime, we avoid the callback verification that is required by the OAuth2 specification.	An application-specific callback URL should be provided with a valid domain name.
Clients do not require SSL/TLS and the secured applications are not exposed over HTTPS.	The boosters are simplified by not requiring certificates generated for each runtime.	In production a secure application should use HTTPS rather than plain HTTP.
The token timeout has been increased to 10 minutes from the default of 1 minute.	Provides a better user experience when working with the command line examples	From a security perspective, the window an attacker would have to guess the access token is extended. It is recommended to keep this window short as it makes it much harder for a potential attacker to guess the current token.

3.6.4. Red Hat SSO realm model

The **`master`** realm is used to secure this booster. There are two pre-configured application client definitions that provide a model for command line clients and the secured REST endpoint.

There are also two pre-configured users in the Red Hat SSO **`master`** realm that can be used to validate various authentication and authorization outcomes: **`admin`** and **`alice`**.

3.6.4.1. Red Hat SSO users

The realm model for the secured boosters includes two users:

admin

The **`admin`** user has a password of **`admin`** and is the realm administrator. This user has full access to the Red Hat SSO administration console, but none of the role mappings that are required to access the secured endpoints. You can use this user to illustrate the behavior of an authenticated, but unauthorized user.

alice

The **`alice`** user has a password of **`password`** and is the canonical application user. This user will demonstrate successful authenticated and authorized access to the secured endpoints. An example representation of the role mappings is provided in this decoded JWT bearer token:

```
{
  "jti": "0073cfaa-7ed6-4326-ac07-c108d34b4f82",
  "exp": 1510162193,
  "nbf": 0,
  "iat": 1510161593,
  "iss": "https://secure-sso-
sso.LOCAL_OPENSHIFT_HOSTNAME/auth/realms/master", (1)
```

```
  "aud": "demoapp",
  "sub": "c0175ccb-0892-4b31-829f-dda873815fe8",
  "typ": "Bearer",
  "azp": "demoapp",
  "nonce": "90ff5d1a-ba44-45ae-a413-50b08bf4a242",
  "auth_time": 1510161591,
  "session_state": "98efb95a-b355-43d1-996b-0abcb1304352",
  "acr": "1",
  "client_session": "5962112c-2b19-461e-8aac-84ab512d2a01",
  "allowed-origins": [
    "*"
  ],
  "realm_access": {
    "roles": [ 2
      "booster-admin"
    ]
  },
  "resource_access": { 3
    "secured-booster-endpoint": {
      "roles": [
        "booster-admin" 4
      ]
    },
    "account": {
      "roles": [
        "manage-account",
        "view-profile"
      ]
    }
  },
  "name": "Alice InChains",
  "preferred_username": "alice", 5
  "given_name": "Alice",
  "family_name": "InChains",
  "email": "alice@keycloak.org"
}
```

1. The **`iss`** field corresponds to the Red Hat SSO realm instance URL that issues the token. This must be configured in the secured endpoint deployments in order for the token to be verified.

2. The **`roles`** object provides the roles that have been granted to the user at the global realm level. In this case **`alice`** has been granted the **`booster-admin`** role. We will see that the secured endpoint will look to the realm level for authorized roles.

3. The **`resource_access`** object contains resource specific role grants. Under this object you will find an object for each of the secured endpoints.

4. The **`resource_access.secured-booster-endpoint.roles`** object contains the roles granted to **`alice`** for the **`secured-booster-endpoint`** resource.

5. The **`preferred_username`** field provides the username that was used to generate the access token.

3.6.4.2. The application clients

The OAuth 2.0 specification allows you to define a role for application clients that access secured resources on behalf of resource owners. The **`master`** realm has the following application clients defined:

demoapp

This is a **`confidential`** type client with a client secret that is used to obtain an access token that contains grants for the **`alice`** user which enable **`alice`** to access the Thorntail, Eclipse Vert.x, Node.js and Spring Boot based REST booster deployments.

secured-booster-endpoint

The **`secured-booster-endpoint`** is a bearer-only type of client that requires a **`booster-admin`** role for accessing the associated resources, specifically the Greeting service.

3.6.5. Thorntail SSO adapter configuration

The SSO adapter is the *client side*, or client to the SSO server, component that enforces security on the web resources. In this specific case, it is the greeting service.

In Thorntail, the security configuration breaks down into two notable assets:

- The **`web.xml`** configuration to enact the security for the service
- The **`keycloak.json`** configuration for the keycloak adapter.

Enacting Security using `web.xml`

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
  <security-constraint>
    <web-resource-collection>
      <url-pattern>/api/greeting</url-pattern> (1)
    </web-resource-collection>
    <auth-constraint>
      <role-name>booster-admin</role-name> (2)
    </auth-constraint>
  </security-constraint>

  <login-config>
    <auth-method>KEYCLOAK</auth-method> (3)
  </login-config>

  <security-role>
    <role-name>booster-admin</role-name>
  </security-role>
</web-app>
```

(1) The web context that is to be secured.

(2) The role needed to access the endpoint.

(3) Using keycloak as the security provider.

Enacting Security in Keycloak Adapter using `keycloak.json`

```
{
  "realm": "master", (1)
  "resource": "secured-booster-endpoint", (2)
  "realm-public-key": "...", (3)
  "auth-server-url": "${sso.auth.server.url}", (4)
  "ssl-required": "external",
  "disable-trust-manager": true,
  "bearer-only": true, (5)
  "use-resource-role-mappings": true
}
```

1. The security realm to be used.
2. The actual keycloak *client* configuration.
3. PEM format of the realm public key. You can obtain this from the administration console.
4. The address of the Red Hat SSO server (Interpolation at build time).
5. If enabled the adapter will not attempt to authenticate users, but only verify bearer tokens.

The **`web.xml`** enables keycloak and enforces protection of the Greeting service web resource endpoint. The **`keycloak.json`** configures the security adapter to interact with Red Hat SSO.

3.6.6. Deploying the Secured booster to Single-node OpenShift Cluster

3.6.6.1. Getting the Fabric8 Launcher tool URL and credentials

You need the Fabric8 Launcher tool URL and user credentials to create and deploy boosters on Single-node OpenShift Cluster. This information is provided when the Single-node OpenShift Cluster is started.

Prerequisites

- The Fabric8 Launcher tool installed, configured, and running. For more information, see the Install and Configure the Fabric8 Launcher Tool guide.

Procedure

1. Navigate to the console where you started Single-node OpenShift Cluster.
2. Check the console output for the URL and user credentials you can use to access the running Fabric8 Launcher:

   Example Console Output from a Single-node OpenShift Cluster Startup

   ```
   ...
   -- Removing temporary directory ... OK
   -- Server Information ...
      OpenShift server started.
      The server is accessible via web console at:
          https://192.168.42.152:8443

      You are logged in as:
   ```

```
        User:     developer
        Password: developer

    To login as administrator:
        oc login -u system:admin
```

3.6.6.2. Creating the Secured booster using Fabric8 Launcher

Prerequisites

- The URL and user credentials of your running Fabric8 Launcher instance. For more information, see Section 3.6.6.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

- Navigate to the Fabric8 Launcher URL in a browser and log in.
- Follow the on-screen instructions to create your booster in Thorntail. When asked about which deployment type, select *I will build and run locally.*
- Follow on-screen instructions.
  When done, click the **Download as ZIP file** button and store the file on your hard drive.

3.6.6.3. Authenticating the `oc` CLI client

To work with boosters on Single-node OpenShift Cluster using the **oc** command-line client, you need to authenticate the client using the token provided by the Single-node OpenShift Cluster web interface.

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.6.6.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Single-node OpenShift Cluster URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your Single-node OpenShift Cluster account.

   ```
   $ oc login OPENSHIFT_URL --token=MYTOKEN
   ```

3.6.6.4. Deploying the Secured booster using the `oc` CLI client

Prerequisites

- The booster application created using the Fabric8 Launcher tool on a Single-node OpenShift Cluster. For more information, see Section 3.6.6.2, “Creating the Secured booster using Fabric8 Launcher”.
- Your Fabric8 Launcher URL.
- The **oc** client authenticated. For more information, see Section 3.6.6.3, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.
4. Deploy the Red Hat SSO server using the **`service.sso.yaml`** file from your booster ZIP file:

```
$ oc create -f service.sso.yaml
```

5. Use Maven to start the deployment to Single-node OpenShift Cluster.

```
$ mvn clean fabric8:deploy -Popenshift -DskipTests \
      -DSSO_AUTH_SERVER_URL=$(oc get route secure-sso -o
jsonpath='{"https://"}{.spec.host}{"/auth\n"}')
```

   This command uses the Fabric8 Maven Plugin to launch the S2I process on Single-node OpenShift Cluster and to start the pod.

This process generates the uberjar file as well as the OpenShift resources and deploys them to the current project on your Single-node OpenShift Cluster server.

3.6.7. Deploying the Secured booster to OpenShift Container Platform

In addition to the Single-node OpenShift Cluster, you can create and deploy the booster on OpenShift Container Platform with only minor differences. The most important difference is that you need to create the booster application on Single-node OpenShift Cluster before you can deploy it with OpenShift Container Platform.

Prerequisites

- The booster created using Single-node OpenShift Cluster.

3.6.7.1. Authenticating the oc CLI client

To work with boosters on OpenShift Container Platform using the **oc** command-line client, you need to authenticate the client using the token provided by the OpenShift Container Platform web interface.

Prerequisites

- An account at OpenShift Container Platform.

Procedure

1. Navigate to the OpenShift Container Platform URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Container Platform account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.6.7.2. Deploying the Secured booster using the oc CLI client

Prerequisites

- The booster application created using the Fabric8 Launcher tool on a Single-node OpenShift Cluster.
- The **oc** client authenticated. For more information, see Section 3.6.7.1, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new OpenShift project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.
4. Deploy the Red Hat SSO server using the **`service.sso.yaml`** file from your booster ZIP file:

```
$ oc create -f service.sso.yaml
```

5. Use Maven to start the deployment to OpenShift Container Platform.

```
$ mvn clean fabric8:deploy -Popenshift -DskipTests \
      -DSSO_AUTH_SERVER_URL=$(oc get route secure-sso -o
jsonpath='{"https://"}{.spec.host}{"/auth\n"}')
```

This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift Container Platform and to start the pod.

This process generates the uberjar file as well as the OpenShift resources and deploys them to the current project on your OpenShift Container Platform server.

3.6.8. Authenticating to the Secured booster API endpoint

The Secured booster provides a default HTTP endpoint that accepts **GET** requests if the caller is authenticated and authorized. The client first authenticates against the Red Hat SSO server and then performs a **GET** request against the Secured booster using the access token returned by the authentication step.

3.6.8.1. Getting the Secured booster API endpoint

When using a client to interact with the booster, you must specify the Secured booster endpoint, which is the *PROJECT_ID* service.

Prerequisites

- The Secured booster deployed and running.
- The **oc** client authenticated.

Procedure

1. In a terminal application, execute the **`oc get routes`** command.
   A sample output is shown in the following table:

Example 3.1. List of Secured endpoints

Name	Host/Port	Path	Services	Port	Termination
secure-sso	secure-sso-myproject.LOCAL_OPENSHIFT_HOSTNAME		secure-sso	<all>	passthrough

Name	Host/Port	Path	Services	Port	Termination
PROJECT_ID	PROJECT_ID-myproject.LOCAL_OPENSHIFT_HOSTNAME		PROJECT_ID	<all>	
sso	sso-myproject.LOCAL_OPENSHIFT_HOSTNAME		sso	<all>	

In the above example, the booster endpoint would be **`http://PROJECT_ID-myproject.LOCAL_OPENSHIFT_HOSTNAME`**. **`PROJECT_ID`** is based on the name you entered when generating your booster using developers.redhat.com/launch or the Fabric8 Launcher tool.

3.6.8.2. Authenticating HTTP requests using the command line

Request a token by sending a HTTP POST request to the Red Hat SSO server. In the following example, the jq CLI tool is used to extract the token value from the JSON response.

Prerequisites

- The secured booster endpoint URL. For more information, see Section 3.6.8.1, “Getting the Secured booster API endpoint”.
- The **`jq`** command-line tool (optional). To download the tool and for more information, see https://stedolan.github.io/jq/.

Procedure

1. Request an access token with **`curl`**, the credentials, and **`<SSO_AUTH_SERVER_URL>`** and extract the token from the response with the **`jq`** command:

```
curl -sk -X POST
https://<SSO_AUTH_SERVER_URL>/auth/realms/master/protocol/openid-
connect/token \
  -d grant_type=password \
  -d username=alice\
  -d password=password \
  -d client_id=demoapp \
  -d client_secret=1daa57a2-b60e-468b-a3ac-25bd2dc2eadc \
  | jq -r '.access_token'

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRek1nbXhZMUhrQnpx
TnR0SnkwMm5jNTNtMGNiWDQxV1hNSTU1MFo4MGVBIn0.eyJqdGkiOiI0NDA3YTliNC04
YWRhLTRlMTctODQ2ZS03YjI5MjMyN2RmYTIiLCJleHAiOjE1MDc3OTM3ODcsIm5iZiI6
MCwiaWF0IjoxNTA3NzkzNzI3LCJpc3MiOiJodHRwczovL3NlY3VyZS1zc28tc3NvLWRl
```

```
bW8uYXBwcy5jYWZlLWJhYmUub3JnL2F1dGgvcmVhbG1zL21hc3RlciIsImF1ZCI6ImRl
bW9hcHAiLCJzdWIiOiJjMDE3NWNjYi0wODkyLTRiMzEtODI5Zi1kZGE4NzM4MTVmZTgi
LCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJkZW1vYXBwIiwiYXV0aF90aW1lIjowLCJzZXNz
aW9uX3N0YXRlIjoiMDFjOTkzNGQtNmZmOS00NWYzLWJkNWUtMTU4NDI5ZDZjNDczIiwi
YWNyIjoiMSIsImNsaWVudF9zZXNzaW9uIjoiMzM3Yzk0MTYtYTdlZS00ZWUzLThjZWQt
ODhlODI0MGJjNTAyIiwiYWxsb3dlZC1vcmlnaW5zIjpbIioiXSwicmVhbG1fYWNjZXNz
Ijp7InJvbGVzIjpbImJvb3N0ZXItYWRtaW4iXX0sInJlc291cmNlX2FjY2VzcyI6eyJz
ZWN1cmVkLWJvb3N0ZXItZW5kcG9pbnQiOnsicm9sZXMiOlsiYm9vc3Rlci1hZG1pbiJd
fSwiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsInZpZXctcHJvZmls
ZSJdfX0sIm5hbWUiOiJBbGljZSBJbkNoYWlucyIsInByZWZlcnJlZF91c2VybmFtZSI6
ImFsaWNlIiwiZ2l2ZW5fbmFtZSI6IkFsaWNlIiwiZmFtaWx5X25hbWUiOiJJbkNoYWlu
cyIsImVtYWlsIjoiYWxpY2VAa2V5Y2xvYWsub3JnIn0.mjmZe37enHpigJv0BGuIitOj
-
kfMLPNwYzNd3n0Ax4Nga7KpnfytGyuPSvR4KAG8rzkfBNN9klPYdy7pJEeYlfmnFUkM4
EDrZYgn4qZAznP1Wzy1RfVRdUFi0-
GqFTMPb37o5HRldZZ09QljX_j3GHnoMGXRtYW9RZN4eKkYkcz9hRwgfJoTy2CuwFqeJw
ZYUyXifrfA-JoTr0UmSUed-0NMksGrtJjjPggUGS-
qOn6OgKcmN2vaVAQlxW32y53JqUXctfLQ6DhJzIMYTmOflIPy0sgG1mG7sovQhw1xTg0
vTjdx8zQ-EJcexkj7IivRevRZsslKgqRFWs67jQAFQA
```

<SSO_AUTH_SERVER_URL> is the url of the **secure-sso** service.

The attributes, such as **username**, **password**, and **client_secret** are usually kept secret, but the above command uses the default provided credentials with this booster for demonstration purpose.

If you do not want to use **jq** to extract the token, you can run just the **curl** command and manually extract the access token.

NOTE

The **-sk** option tells curl to ignore failures resulting from self-signed certificates. Do not use this option in a production environment. On macOS, you must have **curl** version **7.56.1** or greater installed. It must also be built with OpenSSL.

1. Invoke the Secured service. Attach the access (bearer) token to the HTTP headers:

```
$ curl -v -H "Authorization: Bearer <TOKEN>"
http://<SERVICE_HOST>/api/greeting

{
    "content": "Hello, World!",
    "id": 2
}
```

Example 3.2. A sample GET Request Headers with an Access (Bearer) Token

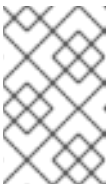
```
> GET /api/greeting HTTP/1.1
> Host: <SERVICE_HOST>
> User-Agent: curl/7.51.0
> Accept: */*
> Authorization: Bearer <TOKEN>
```

<SERVICE_HOST> is the URL of the secured booster endpoint. For more information, see Section 3.6.8.1, “Getting the Secured booster API endpoint”.

2. Verify the signature of the access token.
   The access token is a JSON Web Token, so you can decode it using the JWT Debugger:

   a. In a web browser, navigate to the JWT Debugger website.

   b. Select **RS256** from the *Algorithm* drop down menu.



NOTE

Make sure the web form has been updated after you made the selection, so it displays the correct RSASHA256(…) information in the Signature section. If it has not, try switching to HS256 and then back to RS256.

   c. Paste the following content in the topmost text box into the *VERIFY SIGNATURE* section:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoETnPmN55xBJjRzN/cs30
OzJ9olkteLVNRjzdTxFOyRtS2ovDfzdhhO9XzUcTMbIsCOAZtSt8K+6yvBXypOSYv
I75EUdypmkcK1KoptqY5KEBQ1KwhWuP7IWQ0fshUwD6jI1QWDfGxfM/h34FvEn/0t
J71xN2P8TI2YanwuDZgosdobx/PAvlGREBGuk4BgmexTOkAdnFxIUQcCkiEZ2C41u
CrxiS4CEe5OX91aK9HKZV4ZJX6vnqMHmdDnsMdO+UFtxOBYZio+a1jP4W3d7J5fGe
iOaXjQCOpivKnP2yU2DPdWmDMyVb67l8DRA+jh0OJFKZ5H2fNgE3II59vdsRwIDAQ
AB
-----END PUBLIC KEY-----
```

NOTE

This is the master realm public key from the Red Hat SSO server deployment of the Secured booster.

   d. Paste the **token** output from the client output into the *Encoded* box.
   The *Signature Verified* sign is displayed on the debugger page.

3.6.8.3. Authenticating HTTP requests using the web interface

In addition to the HTTP API, the secured endpoint also contains a web interface to interact with.

The following procedure is an exercise for you to see how security is enforced, how you authenticate, and how you work with the authentication token.

Prerequisites

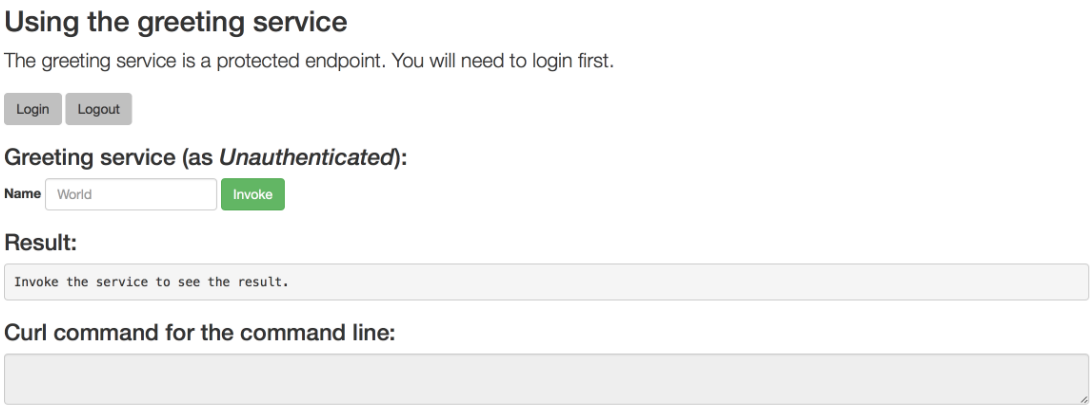
- The secured endpoint URL. For more information, see Section 3.6.8.1, “Getting the Secured booster API endpoint”.

Procedure

1. In a web browser, navigate to the endpoint URL.
2. Perform an unauthenticated request:

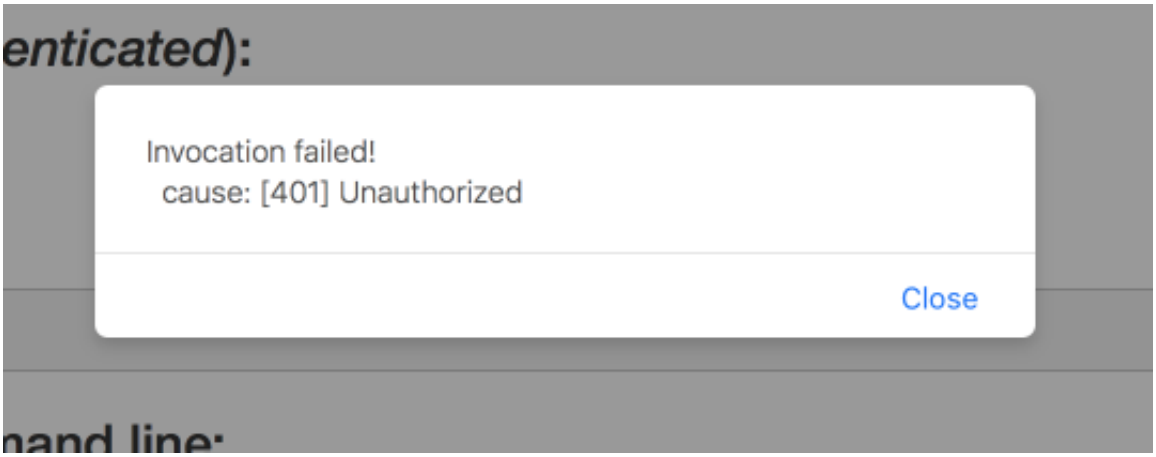
a. Click the *Invoke* button.

Figure 3.1. Unauthenticated Secured Booster Web Interface



The services responds with an **`HTTP 401 Unauthorized`** status code.

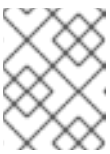
Figure 3.2. Unauthenticated Error Message



3. Perform an authenticated request as a user:

a. Click the *Login* button to authenticate against Red Hat SSO. You will be redirected to the SSO server.

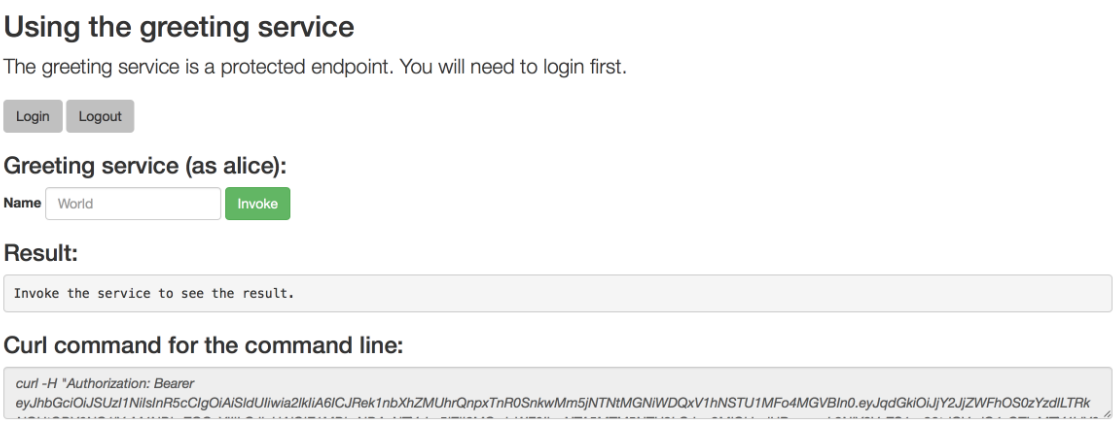
b. Log in as the *Alice* user. You will be redirected back to the web interface.



NOTE

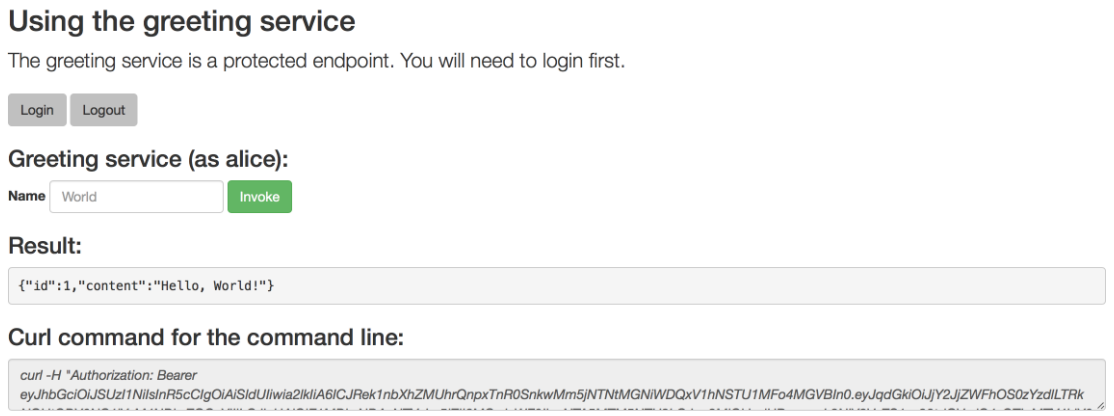
You can see the access (bearer) token in the command line output at the bottom of the page.

Figure 3.3. Authenticated Secured Booster Web Interface (as Alice)



c. Click *Invoke* again to access the Greeting service.
Confirm that there is no exception and the JSON response payload is displayed. This means the service accepted your access (bearer) token and you are authorized access to the Greeting service.

Figure 3.4. The Result of an Authenticated Greeting Request (as Alice)



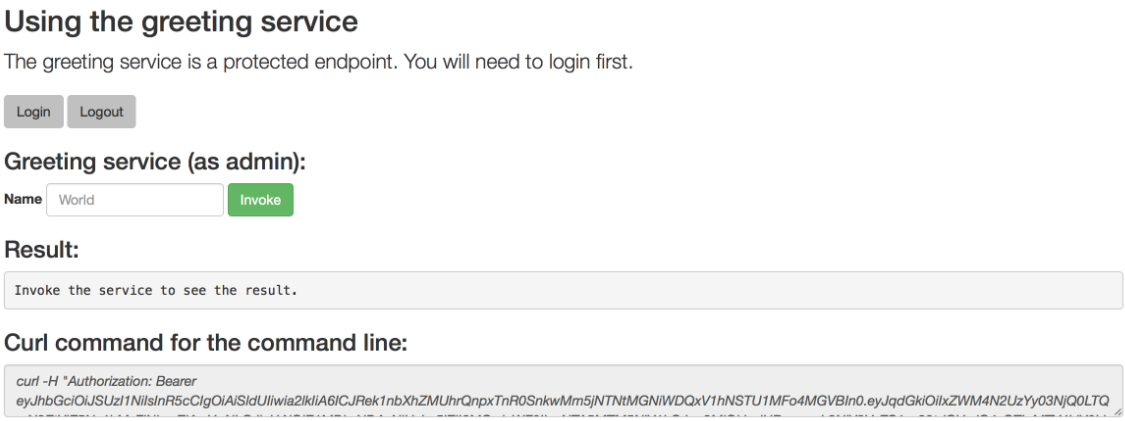
d. Log out.

4. Perform an authenticated request as an admininstrator:

a. Click the *Invoke* button.
Confirm that this sends an unauthenticated request to the Greeting service.

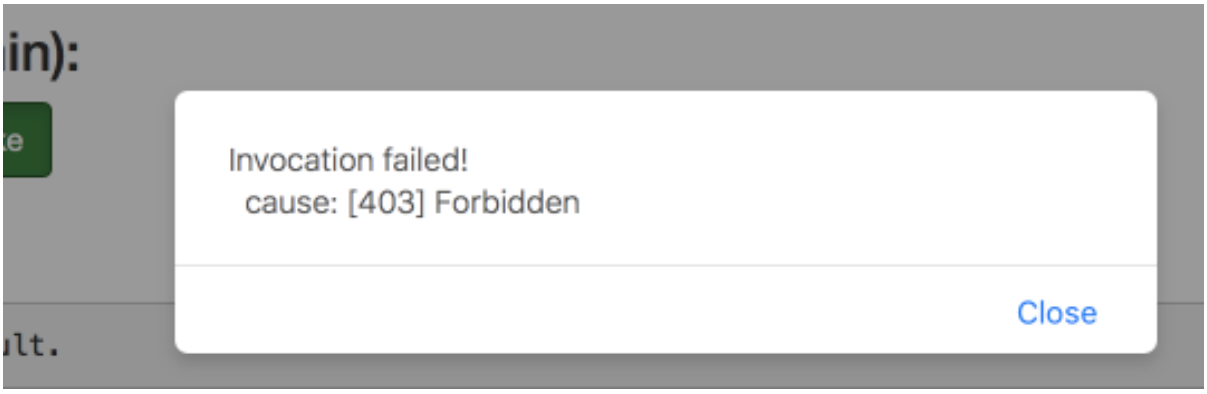
b. Click the *Login* button and log in as the *admin* user.

Figure 3.5. Authenticated Secured Booster Web Interface (as admin)

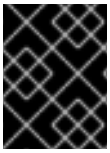


5. Click the *Invoke* button.
   The service responds with an **`HTTP 403 Forbidden`** status code because the *admin* user is not authorized to access the Greeting service.

Figure 3.6. Unauthorized Error Message



3.6.9. Running the Thorntail Secured booster integration tests



IMPORTANT

The **`keycloak-authz-client`** library for Thorntail is provided as a Technology Preview.

Prerequisites

- The **oc** client authenticated.

PROCEDURE

Executing integration tests removes all existing instances of the booster application from the target OpenShift project. To avoid accidentally removing your booster application, ensure that you create and select a separate OpenShift project to execute the tests.

1. In a terminal application, navigate to the directory with your project.
2. Deploy the Red Hat SSO server:

```
oc apply -f service.sso.yaml
```

3. Wait until the Red Hat SSO server is ready. Go to the Web console or view the output of **`oc get pods`** to check if the pod is ready.
4. Execute the integration tests. Provide the URL of the Red Hat SSO server as a parameter:

```
$ mvn clean verify -Popenshift,openshift-it -
DSSO_AUTH_SERVER_URL=$(oc get route secure-sso -o
jsonpath='{"https://"}{.spec.host}{"/auth\n"}')
```

5. Once the tests are finished, remove the Red Hat SSO server:

```
oc delete -f service.sso.yaml
```

3.6.10. Secured SSO resources

Follow the links below for additional information on the principles behind the OAuth2 specification and on securing your applications using Red Hat SSO and Keycloak:

- Aaron Parecki: OAuth2 Simplified
- Red Hat SSO 7.1 Documentation
- Keycloak 3.2 Documentation
- Secured Mission - Spring Boot Booster
- Secured Mission - Eclipse Vert.x Booster
- Secured Mission - Node.js Booster

3.7. CACHE MISSION - THORNTAIL BOOSTER

Limitation: Run this booster on a Single-node OpenShift Cluster. You can also use a manual workflow to deploy this booster to OpenShift Online Pro and OpenShift Container Platform. This booster is not currently available on OpenShift Online Starter.

Mission proficiency level: **Advanced**.

The Cache mission demonstrates how to use a cache to increase the response time of applications.

This mission shows you how to:

- Deploy a cache to OpenShift.
- Use a cache within an application.

3.7.1. How caching works and when you need it

Caches allows you to store information and access it for a given period of time. You can access information in a cache faster or more reliably than repeatedly calling the original service. A disadvantage of using a cache is that the cached information is not up to date. However, that problem can be reduced by setting an *expiration* or TTL (time to live) on each value stored in the cache.

Example 3.3. Caching example

Assume you have two applications: *service1* and *service2*:

- *Service1* depends on a value from *service2*.
  - If the value from *service2* infrequently changes, *service1* could cache the value from *service2* for a period of time.
  - Using cached values can also reduce the number of times *service2* is called.
- If it takes *service1* 500 ms to retrieve the value directly from *service2*, but 100 ms to retrieve the cached value, *service1* would save 400 ms by using the cached value for each cached call.
- If *service1* would make uncached calls to *service2* 5 times per second, over 10 seconds, that would be 50 calls.
- If *service1* started using a cached value with a TTL of 1 second instead, that would be reduced to 10 calls over 10 seconds.

How the Cache mission works

1. The *cache*, *cute name*, and *greeting* services are deployed and exposed.
2. User accesses the web frontend of the *greeting* service.
3. User invokes the *greeting* HTTP API using a button on the web frontend.
4. The *greeting* service depends on a value from the *cute name* service.
   - The *greeting* service first checks if that value is stored in the *cache* service. If it is, then the cached value is returned.
   - If the value is not cached, the *greeting* service calls the *cute name* service, returns the value, and stores the value in the *cache* service with a TTL of 5 seconds.
5. The web front end displays the response from the *greeting* service as well as the total time of the operation.
6. User invokes the service multiple times to see the difference between cached and uncached operations.
   - Cached operations are significantly faster than uncached operations.
   - User can force the cache to be cleared before the TTL expires.

3.7.2. Viewing the booster source code and README

Prerequisites

One of the following:

- Access to developers.redhat.com/launch
- Fabric8 Launcher installed on a Single-node OpenShift Cluster

Procedure

1. Use the Fabric8 Launcher tool to generate your own version of the booster.
2. View the generated GitHub repository or download and extract the ZIP file that contains the booster source code.

Additional resources

- Using developers.redhat.com/launch
- Using the Fabric8 Launcher tool on a Single-node OpenShift Cluster

3.7.3. Deploying the Cache booster to OpenShift Online

Use one of the following options to execute the Cache booster on OpenShift Online.

- Use developers.redhat.com/launch
- Use the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using developers.redhat.com/launch provides an automated booster deployment workflow that executes the **oc** commands for you.

3.7.3.1. Deploying the booster using developers.redhat.com/launch

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the developers.redhat.com/launch URL in a browser and log in.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.7.3.2. Authenticating the oc CLI client

To work with boosters on OpenShift Online using the **oc** command-line client, you need to authenticate the client using the token provided by the OpenShift Online web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the OpenShift Online URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.
4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.
5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.7.3.3. Deploying the Cache booster using the `oc` CLI client

Prerequisites

- The booster application created using developers.redhat.com/launch. For more information, see Section 3.7.3.1, “Deploying the booster using developers.redhat.com/launch”.
- The **oc** client authenticated. For more information, see Section 3.7.3.2, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.
4. Deploy the cache service.

```
$ oc apply -f service.cache.yml
```

5. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

6. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                  READY     STATUS        RESTARTS
```

```
AGE
cache-server-123456789-aaaaa              1/1       Running     0
8m
MY_APP_NAME-cutename-1-bbbbb       1/1       Running     0
4m
MY_APP_NAME-cutename-s2i-1-build   0/1       Completed   0
7m
MY_APP_NAME-greeting-1-ccccc       1/1       Running     0
3m
MY_APP_NAME-greeting-s2i-1-build   0/1       Completed   0
3m
```

Your 3 pods should have a status of **`Running`** once they are fully deployed and started.

7. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                  HOST/PORT
PATH      SERVICES         PORT      TERMINATION
MY_APP_NAME-cutename   MY_APP_NAME-cutename-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME             MY_APP_NAME-cutename
8080                    None
MY_APP_NAME-greeting   MY_APP_NAME-greeting-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME             MY_APP_NAME-greeting
8080                    None
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-greeting-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the greeting service.

3.7.4. Deploying the Cache booster to Single-node OpenShift Cluster

Use one of the following options to execute the Cache booster locally on Single-node OpenShift Cluster:

- Using Fabric8 Launcher
- Using the **oc** CLI client

Although each method uses the same **oc** commands to deploy your application, using Fabric8 Launcher provides an automated booster deployment workflow that executes the **oc** commands for you.

3.7.4.1. Getting the Fabric8 Launcher tool URL and credentials

You need the Fabric8 Launcher tool URL and user credentials to create and deploy boosters on Single-node OpenShift Cluster. This information is provided when the Single-node OpenShift Cluster is started.

Prerequisites

- The Fabric8 Launcher tool installed, configured, and running. For more information, see the Install and Configure the Fabric8 Launcher Tool guide.

Procedure

1. Navigate to the console where you started Single-node OpenShift Cluster.
2. Check the console output for the URL and user credentials you can use to access the running Fabric8 Launcher:

Example Console Output from a Single-node OpenShift Cluster Startup

```
...
-- Removing temporary directory ... OK
-- Server Information ...
   OpenShift server started.
   The server is accessible via web console at:
       https://192.168.42.152:8443

   You are logged in as:
       User:     developer
       Password: developer

   To login as administrator:
       oc login -u system:admin
```

3.7.4.2. Deploying the booster using the Fabric8 Launcher tool

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.7.4.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Fabric8 Launcher URL in a browser.
2. Follow on-screen instructions to create and launch your booster in Thorntail.

3.7.4.3. Authenticating the `oc` CLI client

To work with boosters on Single-node OpenShift Cluster using the **oc** command-line client, you need to authenticate the client using the token provided by the Single-node OpenShift Cluster web interface.

Prerequisites

- The URL of your running Fabric8 Launcher instance and the user credentials of your Single-node OpenShift Cluster. For more information, see Section 3.7.4.1, “Getting the Fabric8 Launcher tool URL and credentials”.

Procedure

1. Navigate to the Single-node OpenShift Cluster URL in a browser.
2. Click on the question mark icon in the top right-hand corner of the Web console, next to your user name.
3. Select *Command Line Tools* in the drop-down menu.

4. Find the text box that contains the **`oc login …`** command with the hidden token, and click the button next to it to copy its content to your clipboard.

5. Paste the command into a terminal application. The command uses your authentication token to authenticate your **oc** CLI client with your Single-node OpenShift Cluster account.

```
$ oc login OPENSHIFT_URL --token=MYTOKEN
```

3.7.4.4. Deploying the Cache booster using the `oc` CLI client

Prerequisites

- The booster application created using Fabric8 Launcher tool on a Single-node OpenShift Cluster. For more information, see Section 3.7.4.2, “Deploying the booster using the Fabric8 Launcher tool”.
- Your Fabric8 Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 3.7.4.3, “Authenticating the **oc** CLI client”.

Procedure

1. Clone your project from GitHub.

```
$ git clone git@github.com:USERNAME/MY_PROJECT_NAME.git
```

   Alternatively, if you downloaded a ZIP file of your project, extract it.

```
$ unzip MY_PROJECT_NAME.zip
```

2. Create a new project.

```
$ oc new-project MY_PROJECT_NAME
```

3. Navigate to the root directory of your booster.

4. Deploy the cache service.

```
$ oc apply -f service.cache.yml
```

5. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

6. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                READY     STATUS      RESTARTS
AGE
cache-server-123456789-aaaaa              1/1       Running     0
8m
MY_APP_NAME-cutename-1-bbbbb       1/1       Running     0
```

```
4m
MY_APP_NAME-cutename-s2i-1-build   0/1       Completed   0
7m
MY_APP_NAME-greeting-1-ccccc       1/1       Running     0
3m
MY_APP_NAME-greeting-s2i-1-build   0/1       Completed   0
3m
```

Your 3 pods should have a status of **`Running`** once they are fully deployed and started.

7. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                 HOST/PORT
PATH      SERVICES        PORT      TERMINATION
MY_APP_NAME-cutename   MY_APP_NAME-cutename-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME             MY_APP_NAME-cutename
8080                    None
MY_APP_NAME-greeting   MY_APP_NAME-greeting-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME             MY_APP_NAME-greeting
8080                    None
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **`http://MY_APP_NAME-greeting-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME`** as the base URL to access the greeting service.

3.7.5. Deploying the Cache booster to OpenShift Container Platform

The process of creating and deploying boosters to OpenShift Container Platform is similar to OpenShift Online:

Prerequisites

- The booster created using developers.redhat.com/launch or the Fabric8 Launcher tool.

Procedure

- Follow the instructions in Section 3.7.3, “Deploying the Cache booster to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

3.7.6. Interacting with the unmodified Cache booster

Prerequisites

- Your application deployed

Procedure

1. Navigate to the **`greeting`** service using your browser.
2. Click *Invoke the service* once.

Notice the **`duration`** value is above **`2000`**. Also notice the cache state has changed form **`No cached value`** to **`A value is cached`**.

3. Wait 5 seconds and notice cache state has changed back to **`No cached value`**.
   The TTL for the cached value is set to 5 seconds. When the TTL expires, the value is no longer cached.

4. Click *Invoke the service* once more to cache the value.

5. Click *Invoke the service* a few more times over the course of a few seconds while cache state is **`A value is cached`**.
   Notice a significantly lower **`duration`** value since it is using a cached value. If you click *Clear the cache*, the cache is emptied.

3.7.7. Running the Cache booster integration tests

This booster includes a self-contained set of integration tests. When run inside an OpenShift project, the tests:

- Deploy a test instance of the application to the project.
- Execute the individual tests on that instance.
- Remove all instances of the application from the project when the testing is done.

WARNING

Executing integration tests removes all existing instances of the booster application from the target OpenShift project. To avoid accidentally removing your booster application, ensure that you create and select a separate OpenShift project to execute the tests.

Prerequisites

- The **oc** client authenticated
- An empty OpenShift project

Procedure

Execute the following command to run the integration tests:

```
$ mvn clean verify -Popenshift,openshift-it
```

3.7.8. Caching resources

More background and related information on caching can be found here:

- Cache Mission - Spring Boot Booster

- Cache Mission - Eclipse Vert.x Booster
- Cache Mission - Node.js Booster

CHAPTER 4. DEVELOPING AN APPLICATION FOR THE THORNTAIL RUNTIME

This sections contains information about creating your application and configuring the Thorntail runtime for it.

4.1. CREATING A BASIC THORNTAIL APPLICATION

In addition to using a booster, you can create new Thorntail applications from scratch and deploy them to OpenShift.

4.1.1. Creating an application from scratch

Creating a simple Thorntail–based application with a REST endpoint from scratch.

Prerequisites

- JDK 8 or newer installed
- Maven 3.3.x or newer installed

Procedure

1. Create a directory for the application and navigate to it:

```
$ mkdir myApp
$ cd myApp
```

   We recommend you start tracking the directory contents with Git. For more information, see Git tutorial.

2. In the directory, create a **`pom.xml`** file with the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>restful-endpoint</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>Thorntail Example</name>

  <properties>
    <version.thorntail>2.2.0.Final-redhat-00021</version.thorntail>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <failOnMissingWebXml>false</failOnMissingWebXml>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    <fabric8.generator.from>registry.access.redhat.com/redhat-
```

```
openjdk-18/openjdk18-openshift:1.2</fabric8.generator.from>
  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>io.thorntail</groupId>
        <artifactId>bom</artifactId>
        <version>${version.thorntail}</version>
        <scope>import</scope>
        <type>pom</type>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.thorntail</groupId>
      <artifactId>jaxrs</artifactId>
    </dependency>
  </dependencies>

  <build>
    <finalName>restful-endpoint</finalName>
    <plugins>
      <plugin>
        <groupId>io.thorntail</groupId>
        <artifactId>thorntail-maven-plugin</artifactId>
        <version>${version.thorntail}</version>
        <executions>
          <execution>
            <goals>
              <goal>package</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>

  <!-- Specify the repositories containing RHOAR artifacts -->
  <repositories>
    <repository>
      <id>redhat-ga</id>
      <name>Red Hat GA Repository</name>
      <url>https://maven.repository.redhat.com/ga/</url>
    </repository>
  </repositories>

  <pluginRepositories>
    <pluginRepository>
      <id>redhat-ga</id>
      <name>Red Hat GA Repository</name>
      <url>https://maven.repository.redhat.com/ga/</url>
    </pluginRepository>
  </pluginRepositories>
```

```
  <profiles>
    <profile>
      <id>openshift</id>
      <build>
        <plugins>
          <plugin>
            <groupId>io.fabric8</groupId>
            <artifactId>fabric8-maven-plugin</artifactId>
            <executions>
              <execution>
                <goals>
                  <goal>resource</goal>
                  <goal>build</goal>
                </goals>
              </execution>
            </executions>
          </plugin>
        </plugins>
      </build>
    </profile>
  </profiles>
</project>
```

3. Create a directory structure for your application:

```
mkdir -p src/main/java/com/example/rest
```

4. In the **`src/main/java/com/example/rest`** directory, create the source files:

- **`HelloWorldEndpoint.java`** with the class that serves the HTTP endpoint:

```
package com.example.rest;

import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
import javax.ws.rs.GET;
import javax.ws.rs.Produces;

@Path("/hello")
public class HelloWorldEndpoint {

  @GET
  @Produces("text/plain")
  public Response doGet() {
    return Response.ok("Hello from Thorntail!").build();
  }
}
```

- **`RestApplication.java`** with the application context:

```
package com.example.rest;

import javax.ws.rs.core.Application;
import javax.ws.rs.ApplicationPath;
```

```
@ApplicationPath("/rest")
public class RestApplication extends Application {
}
```

5. Execute the application using Maven:

```
$ mvn thorntail:run
```

Results
Accessing the **http://localhost:8080/rest/hello** URL in your browser should return the following message:

```
Hello from Thorntail!
```

After finishing the procedure, there should be a directory on your hard drive with the following contents:

```
myApp
├── pom.xml
└── src
    └── main
        └── java
            └── com
                └── example
                    └── rest
                        ├── HelloWorldEndpoint.java
                        └── RestApplication.java
```

4.1.2. Deploying an application to OpenShift

This procedure shows you how to:

- Build your application and deploy it to OpenShift using the Fabric8 Maven Plugin.
- Use the command line to interact with your application running on OpenShift.

Prerequisites

- The **oc** CLI client installed.
- Maven installed.
- A Maven-based application.

Procedure

1. Log in to your OpenShift instance with the **oc** client.

```
$ oc login ...
```

2. Create a new project.

```
$ oc new-project MY_PROJECT_NAME
```

3. In a terminal application, navigate to the directory containing your application:

```
$ cd myApp
```

4. Use Maven to start the deployment to OpenShift.

```
$ mvn clean fabric8:deploy -Popenshift
```

This command uses the Fabric8 Maven Plugin to launch the S2I process on OpenShift and to start the pod.

5. Check the status of your booster and ensure your pod is running.

```
$ oc get pods -w
NAME                                READY     STATUS      RESTARTS
AGE
MY_APP_NAME-1-aaaaa                 1/1       Running     0
58s
MY_APP_NAME-s2i-1-build             0/1       Completed   0
2m
```

The **MY_APP_NAME-1-aaaaa** pod should have a status of **Running** once it is fully deployed and started. Your specific pod name will vary. The number in the middle will increase with each new build. The letters at the end are generated when the pod is created.

6. Once your booster is deployed and started, determine its route.

Example Route Information

```
$ oc get routes
NAME                 HOST/PORT
PATH      SERVICES         PORT       TERMINATION
MY_APP_NAME          MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME
MY_APP_NAME       8080
```

The route information of a pod gives you the base URL which you use to access it. In the example above, you would use **http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME** as the base URL to access the application.

7. Using **curl** or your browser, verify your application is running in OpenShift.

```
$ curl http://MY_APP_NAME-
MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/rest/hello
Hello from Thorntail!
```

4.2. DEPLOYING AN EXISTING THORNTAIL APPLICATION TO OPENSHIFT

You can easily deploy your existing application to OpenShift using the Fabric8 Maven plugin.

Prerequisites

- A Thorntail–based application

Procedure

1. Add the following profile to the **pom.xml** file in the root directory of your application:

```
<profile>
  <id>openshift</id>
  <build>
    <plugins>
      <plugin>
        <groupId>io.fabric8</groupId>
        <artifactId>fabric8-maven-plugin</artifactId>
        <version>3.5.40</version>
        <executions>
          <execution>
            <goals>
              <goal>resource</goal>
              <goal>build</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <generator>
            <includes>
              <include>thorntail-v2</include>
            </includes>
            <excludes>
              <exclude>webapp</exclude>
            </excludes>
          </generator>
          <enricher>
            <config>
              <thorntail-v2-health-check>
                <path>/</path>
              </thorntail-v2-health-check>
            </config>
          </enricher>
        </configuration>
      </plugin>
    </plugins>
  </build>
</profile>
```

   In this profile, the Fabric8 Maven plugin is invoked for building and deploying the application to OpenShift.

2. Deploy the application to OpenShift according to instructions in Section 4.1.2, “Deploying an application to OpenShift”.

4.3. USING THORNTAIL MAVEN PLUGIN

Thorntail provides a Maven plugin to accomplish most of the work of building uberjar packages.

4.3.1. Thorntail Maven plugin general usage

The Thorntail Maven plugin is used like any other Maven plugin, that is through editing the **pom.xml** file in your application and adding a **<plugin>** section:

```
<plugin>
  <groupId>io.thorntail</groupId>
  <artifactId>thorntail-maven-plugin</artifactId>
  <version>${version.thorntail}</version>
  <executions>
    ...
    <execution>
      <goals>
        ...
      </goals>
      <configuration>
        ...
      </configuration>
    </execution>
  </executions>
</plugin>
```

4.3.2. Thorntail Maven plugin goals

The Thorntail Maven plugin provides several goals:

package

Creates the executable package (see Section 5.2, “Creating an uberjar”).

run

Executes your application in the Maven process. The application is stopped if the Maven build is interrupted, for example when you press Ctrl + C.

start and multistart

Executes your application in a forked process. Generally, it is only useful for running integration tests using a plugin, such as the **maven-failsafe-plugin**. The **multistart** variant allows starting multiple Thorntail–built applications using Maven GAVs to support complex testing scenarios.

stop

Stops any previously started applications.

NOTE

The **stop** goal can only stop applications that were started *in the same Maven execution*.

4.3.3. Thorntail Maven plugin configuration options

The Thorntail Maven plugin accepts the following configuration options:

bundleDependencies

If true, dependencies will be included in the **-thorntail.jar** file. Otherwise, they will be resolved from **$M2_REPO** or the network at runtime.

Property	`swarm.bundleDependencies`
Default	true
Used by	`package`

debug

The port to use for debugging. If set, the swarm process will suspend on start and open a debugger on this port.

Property	`swarm.debug.port`
Default	
Used by	`run`, `start`

environment

A properties-style list of environment variables to use when executing the application.

Property	*none*
Default	
Used by	`multistart`, `run`, `start`

environmentFile

A `.properties` file with environment variables to use when executing the application.

Property	`swarm.environmentFile`
Default	
Used by	`multistart`, `run`, `start`

fractionDetectMode

The mode of fraction detection. The available options are:

- `when_missing`: Runs only when no Thorntail dependencies are found.
- `force`: Always run, and merge any detected fractions with the existing dependencies. Existing dependencies take precedence.
- `never`: Disable fraction detection.

Property	`swarm.detect.mode`

Default	**when_missing**
Used by	**package**, **run**, **start**

fractions

A list of extra fractions to include when auto-detection is used. It is useful for fractions that cannot be detected or user-provided fractions.
The format of specifying a fraction can be: * **group:artifact:version** * **artifact:version** * **artifact**

If no group is provided, **io.thorntail** is assumed.

If no version is provided, the version is taken from the Thorntail BOM for the version of the plugin you are using.

If the value starts with the character **!** a corresponding auto-detected fraction is not installed (unless it's a dependency of any other fraction). In the following example the Undertow fraction is not installed even if your application references a class from the **javax.servlet** package:

```
<plugin>
  <groupId>io.thorntail</groupId>
  <artifactId>thorntail-maven-plugin</artifactId>
  <version>${version.thorntail}</version>
  <executions>
    <execution>
      <goals>
        <goal>package</goal>
      </goals>
      <configuration>
       <fractions>
         <fraction>!undertow</fraction>
       </fractions>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Property	*none*
Default	
Used by	**package**, **run**, **start**

jvmArguments

A list of **<jvmArgument>** elements specifying additional JVM arguments (such as **-Xmx32m**).

Property	**swarm.jvmArguments**
Default	

Used by	**multistart**, **run**, **start**

modules

Paths to a directory containing additional module definitions.

Property	*none*
Default	
Used by	**package**, **run**, **start**

processes

Application configurations to start (see multistart).

Property	*none*
Default	
Used by	**multistart**

properties

See Section 4.3.4, “Thorntail Maven plugin configuration properties”.

Property	*none*
Default	
Used by	**package**, **run**, **start**

propertiesFile

See Section 4.3.4, “Thorntail Maven plugin configuration properties”.

Property	**swarm.propertiesFile**
Default	
Used by	**package**, **run**, **start**

stderrFile

A file path where to store the **stderr** output instead of sending it to the **stderr** output of the launching process.

Property	**swarm.stderr**

Default	
Used by	**run**, **start**

stdoutFile

A file path where to store the **stdout** output instead of sending it to the **stdout** output of the launching process.

Property	**swarm.stdout**
Default	
Used by	**run**, **start**

useUberJar

If true, the **-thorntail.jar** file specified at **${project.build.directory}** is used. This JAR is not created automatically, so make sure you execute the **package** goal first.

Property	**wildfly-swarm.useUberJar**
Default	false
Used by	**run**, **start**

4.3.4. Thorntail Maven plugin configuration properties

Properties can be used to configure the execution and affect the packaging or running of your application.

If you add a **<properties>** or **<propertiesFile>** section to the **<configuration>** of the plugin, the properties are used when executing your application using the **mvn thorntail:run** command. In addition to that, the same properties are added to your ***myapp*-thorntail.jar** file to affect subsequent executions of the uberjar. Any properties loaded from the **<propertiesFile>** override identically-named properties in the **<properties>** section.

Any properties added to the uberjar can be overridden at runtime using the traditional **-Dname=value** mechanism of the **java** binary, or using the YAML-based configuration files.

Only the following properties are added to the uberjar at package time:

- The properties specified outside of the **<properties>** section or the **<propertiesFile>**, whose path starts with one of the following:
  - **jboss.**
  - **wildfly.**
  - **swarm.**

- **maven.**

- The properties that override a property specified in the **<properties>** section or the **<propertiesFile>**.

4.4. FRACTIONS

Thorntail is defined by an unbounded set of capabilities. Each piece of functionality is called a *fraction.* Some fractions provide only access to APIs, such as JAX-RS or CDI; other fractions provide higher-level capabilities, such as integration with RHSSO (Keycloak).

The typical method for consuming Thorntail fractions is through Maven coordinates, which you add to the **pom.xml** file in your application. The functionality the fraction provides is then packaged with your application (see Section 5.2, “Creating an uberjar”).

To enable easier consumption of Thorntail fractions, a bill of materials (BOM) is available. For more information, see Section 4.5, “Using a BOM”.

4.4.1. Auto-detecting fractions

Migrating existing legacy applications to benefit from Thorntail is simple when using fraction auto-detection. If you enable the Thorntail Maven plugin in your application, Thorntail detects which APIs you use, and includes the appropriate fractions at build time.

NOTE

By default, Thorntail only auto-detects if you do not specify any fractions explicitly. This behavior is controlled by the **fractionDetectMode** property. For more information, see the Maven plugin configuration reference.

For example, consider your **pom.xml** already specifies the API **.jar** file for a specification such as JAX-RS:

```
<dependencies>
    <dependency>
      <groupId>org.jboss.spec.javax.ws.rs</groupId>
      <artifactId>jboss-jaxrs-api_2.0_spec</artifactId>
      <version>${version.jaxrs-api}</version>
      <scope>provided</scope>
    </dependency>
</dependencies>
```

Thorntail then includes the **jaxrs** fraction during the build automatically.

Prerequisites

- An existing Maven-based application with a **pom.xml** file.

Procedure

1. Add the **thorntail-maven-plugin** to your **pom.xml** in a **<plugin>** block, with an **<execution>** specifying the **package** goal.

```
<plugins>
```

```
  <plugin>
    <groupId>io.thorntail</groupId>
    <artifactId>thorntail-maven-plugin</artifactId>
    <version>${version.thorntail}</version>
    <executions>
      <execution>
        <id>package</id>
        <goals>
          <goal>package</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
```

2. Perform a normal Maven build:

```
$ mvn package
```

3. Execute the resulting uberjar:

```
$ java -jar ./target/myapp-thorntail.jar
```

Related Information

- Section 4.4, “Fractions”
- Section 5.2, “Creating an uberjar”

4.4.2. Using explicit fractions

When writing your application from scratch, ensure it compiles correctly and uses the correct version of APIs by explicitly selecting which fractions are packaged with it.

Prerequisites

- A Maven-based application with a **pom.xml** file.

Procedure

1. Add the BOM to your **pom.xml**. For more information, see Section 4.5, “Using a BOM”.
2. Add the Thorntail Maven plugin to your **pom.xml**. For more information, see Section 5.2, “Creating an uberjar”.
3. Add one or more dependencies on Thorntail fractions to the **pom.xml** file:

```
<dependencies>
  <dependency>
    <groupId>io.thorntail</groupId>
    <artifactId>jaxrs</artifactId>
  </dependency>
</dependencies>
```

4. Perform a normal Maven build:

```
$ mvn package
```

5. Execute the resulting uberjar:

```
$ java -jar ./target/myapp-thorntail.jar
```

Related Information

- Section 4.4, “Fractions”
- Section 4.4.1, “Auto-detecting fractions”

4.5. USING A BOM

To explicitly specify the Thorntail fractions your application uses, instead of relying on auto-detection, Thorntail includes a set of BOMs (bill of materials) which you can use instead of having to track and update Maven artifact versions in several places.

4.5.1. Thorntail product BOM types

Thorntail is described as *just enough app-server,* which means it consists of multiple pieces. Your application includes only the pieces it needs.

When using the Thorntail product, you can specify the following Maven BOMs:

bom

All fractions available in the product.

bom-certified

All *community* fractions that have been certified against the product. Any fraction used from **`bom-certified`** is unsupported.

4.5.2. Specifying a BOM for in your application

Importing a specific BOM in the **`pom.xml`** file in your application allows you to track all your application dependencies in one place.

NOTE

One shortcoming of importing a Maven BOM import is that it does not handle the configuration on the level of **<pluginManagement>**. When you use the Thorntail Maven Plugin, you must specify the version of the plugin to use.

Thanks to the property you use in your **pom.xml** file, you can easily ensure that your plugin usage matches the release of Thorntail that you are targeting with the BOM import.

```
<plugins>
  <plugin>
    <groupId>io.thorntail</groupId>
    <artifactId>thorntail-maven-plugin</artifactId>
    <version>${version.thorntail}</version>
      ...
  </plugin>
</plugins>
```

Prerequisites

- Your application as a Maven-based project with a **pom.xml** file.

Procedure

1. Include a **bom** artifact in your **pom.xml**.
   Tracking the current version of Thorntail through a property in your **pom.xml** is recommended.

   ```
   <properties>
     <version.thorntail>2.2.0.Final-redhat-00021</version.thorntail>
   </properties>
   ```

   Import BOMs in the **<dependencyManagement>** section. Specify the **<type>pom</type>** and **<scope>import</scope>**.

   ```
   <dependencyManagement>
     <dependencies>
       <dependency>
         <groupId>io.thorntail</groupId>
         <artifactId>bom</artifactId>
         <version>${version.thorntail}</version>
         <type>pom</type>
         <scope>import</scope>
       </dependency>
     </dependencies>
   </dependencyManagement>
   ```

   In the example above, the **bom** artifact is imported to ensure that only stable fractions are available.

   By including the BOMs of your choice in the **<dependencyManagement>** section, you have:

   - Provided version-management for any Thorntail artifacts you subsequently **choose** to use.

- Provided support to your IDE for auto-completing known artifacts when you edit your the **pom.xml** file of your application.

2. Include Thorntail dependencies.
   Even though you imported the Thorntail BOMs in the **<dependencyManagement>** section, your application still has no dependencies on Thorntail artifacts.

   To include Thorntail artifact dependencies based on the capabilities your application, enter the relevant artifacts as **<dependency>** elements:

   NOTE

   You do not have to specify the version of the artifacts because the BOM imported in **<dependencyManagement>** handles that.

```
<dependencies>
  <dependency>
    <groupId>io.thorntail</groupId>
    <artifactId>jaxrs</artifactId>
  </dependency>
  <dependency>
    <groupId>io.thorntail</groupId>
    <artifactId>datasources</artifactId>
  </dependency>
</dependencies>
```

   In the example above, we include explicit dependencies on the **jaxrs** and **datasources** fractions, which will provide transitive inclusion of others, for example **undertow**.

4.6. LOGGING

4.6.1. Enabling logging

Each Thorntail fraction is dependent on the Logging fraction, which means that if you use any Thorntail fraction in your application, logging is automatically enabled on the **INFO** level and higher. If you want to enable logging explicitly, add the Logging fraction to the POM file of your application.

Prerequisites

- A Maven-based application

Procedure

1. Find the **<dependencies>** section in the **pom.xml** file of your application. Verify it contains the following coordinates. If it does not, add them.

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>logging</artifactId>
</dependency>
```

2. If you want to log messages of a level other than **INFO**, launch the application while specifying the **swarm.logging** system property:

```
$ mvn thorntail:run -Dswarm.logging=FINE
```

See the **org.wildfly.swarm.config.logging.Level** class for the list of available levels.

4.6.2. Logging to a file

In addition to the console logging, you can save the logs of your application in a file. Typically, deployments use rotating logs to save disk space.

In Thorntail, logging is configured using system properties. Even though it is possible to use the **-Dproperty=value** syntax when launching your application, it is strongly recommended to configure file logging using the YAML profile files.

Prerequisites

- A Maven-based application with the logging fraction enabled. For more information, see Section 4.6.1, “Enabling logging”.
- A writable directory on your file system.

Procedure

1. Open a YAML profile file of your choice. If you do not know which one to use, open **project-defaults.yml** in the **src/main/resources** directory in your application sources. In the YAML file, add the following section:

```
swarm:
  logging:
```

2. Configure a formatter (optional). The following formatters are configured by default:

PATTERN

Useful for logging into a file.

COLOR_PATTERN

Color output. Useful for logging to the console.

To configure a custom formatter, add a new formatter with a pattern of your choice in the **logging** section. In this example, it is called **LOG_FORMATTER**:

```
pattern-formatters:
  LOG_FORMATTER:
    pattern: "%p [%c] %s%e%n"
```

3. Configure a file handler to use with the loggers. This example shows the configuration of a periodic rotating file handler. Under **logging**, add a **periodic-rotating-file-handlers** section with a new handler.

```
periodic-rotating-file-handlers:
  FILE:
    file:
      path: target/MY_APP_NAME.log
```

```
      suffix: .yyyy-MM-dd
      named-formatter: LOG_FORMATTER
      level: INFO
```

Here, a new handler named **`FILE`** is created, logging events of the **`INFO`** level and higher. It logs in the **`target`** directory, and each log file is named **`MY_APP_NAME.log`** with the suffix **`.yyyy-MM-dd`**. Thorntail automatically parses the log rotation period from the suffix, so ensure you use a format compatible with the **`java.text.SimpleDateFormat`** class.

4. Configure the root logger.
   The root logger is by default configured to use the **`CONSOLE`** handler only. Under **`logging`**, add a **`root-logger`** section with the handlers you wish to use:

```
root-logger:
  handlers:
  - CONSOLE
  - FILE
```

   Here, the **`FILE`** handler from the previous step is used, along with the default console handler.

Below, you can see the complete logging configuration section:

The logging section in a YAML configuration profile

```
swarm:
  logging:
    pattern-formatters:
      LOG_FORMATTER:
        pattern: "CUSTOM LOG FORMAT %p [%c] %s%e%n"
    periodic-rotating-file-handlers:
      FILE:
        file:
          path: path/to/your/file.log
        suffix: .yyyy-MM-dd
        named-formatter: LOG_FORMATTER
    root-logger:
      handlers:
      - CONSOLE
      - FILE
```

4.7. CONFIGURING A THORNTAIL APPLICATION

You can configure numerous options with applications built with Thorntail. For most options, reasonable defaults are already applied, so you do not have to change any options unless you explicitly want to.

This reference is a complete list of all configurable items, grouped by the fraction that introduces them. Only the items related to the fractions that your application uses are relevant to you.

4.7.1. System properties

Using system properties for configuring your application is advantageous for experimenting, debugging, and other short-term activities.

4.7.1.1. Application configuration using system properties

Configuration properties are presented using dotted notation, and are suitable for use as Java system property names, which your application consumes through explicit setting in the Maven plugin configuration, or through the command line when your application is being executed.

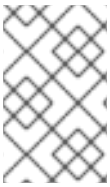
Any property that has the *KEY* parameter in its name indicates that you must supply a key or identifier in that segment of the name.

Configuration of items with the KEY parameter
A configuration item documented as **`swarm.undertow.servers.KEY.default-host`** indicates that the configuration applies to a particular named server.

In practical usage, the property would be, for example, **`swarm.undertow.servers.default.default-host`** for a server known as **`default`**.

4.7.1.2. Setting system properties using the Maven plugin

Setting properties using the Maven plugin is useful for temporarily changing a configuration item for a single execution of your Thorntail application.



NOTE

Even though the configuration in the POM file of your application is persistent, it is not recommended to use it for long-term configuration of your application. Instead, use the YAML configuration files.

If you want to set explicit configuration values as defaults through the Maven plugin, add a **`<properties>`** section to the **`<configuration>`** block of the plugin in the **`pom.xml`** file in your application.

Prerequisites

- Your Thorntail-based application with a POM file

Procedure

1. In the POM file of your application, locate the configuration you want to modify.
2. Insert a block with configuration of the **`io.thorntail:thorntail-maven-plugin`** artifact, for example:

```
<build>
  <plugins>
    <plugin>
      <groupId>io.thorntail</groupId>
      <artifactId>thorntail-maven-plugin</artifactId>
      <version>{version}</version>
      <configuration>
        <properties>
          <swarm.bind.address>127.0.0.1</swarm.bind.address>
          <java.net.preferIPv4Stack>true</java.net.preferIPv4Stack>
        </properties>
      </configuration>
```

```
    </plugin>
  </plugins>
</build>
```

In the example above, the **`swarm.bind.address`** property is set to **`127.0.0.1`** and the **`java.net.preferIPv4Stack`** property is set to **`true`**.

4.7.1.3. Setting system properties using the command line

Setting properties using the Maven plugin is useful for temporarily changing a configuration item for a single execution of your Thorntail application.

You can customize an environment-specific setting or experiment with configuration items before setting them in a YAML configuration file.

To use a property on the command line, pass it as a command-line parameter to the Java binary:

Prerequisites

- A JAR file with your application

Procedure

1. In a terminal application, navigate to the directory with your application JAR file.
2. Execute your application JAR file using the Java binary and specify the property and its value:

```
$ java -Dswarm.bind.address=127.0.0.1 -jar myapp-thorntail.jar
```

In this example, you assing the value **`127.0.0.1`** to the property called **`swarm.bind.address`**.

4.7.1.4. Specifying JDBC drivers for hollow JARs

When executing a hollow JAR, you can specify a JDBC Driver JAR using the **`swarm.classpath`** property. This way, you do not need to package the driver in the hollow JAR.

The **`swarm.classpath`** property accepts one or more paths to JAR files separated by **`;`** (a semicolon). The specified JAR files are added to the classpath of the application.

Prerequisites

- A JAR file with your application

Procedure

1. In a terminal application, navigate to the directory with your application JAR file.
2. Execute your application JAR file using the Java binary and specify the JDBC driver:

```
$ java -Dswarm.classpath=./h2-1.4.196.jar -jar microprofile-jpa-
hollow-thorntail.jar example-jpa-jaxrs-cdi.war
```

4.7.2. YAML files

YAML is the preferred method for long-term configuration of your application. In addition to that, the YAML strategy provides grouping of environment-specific configurations, which you can selectively enable when executing the application.

4.7.2.1. The general YAML file format

The Thorntail configuration item names correspond to the YAML configuration structure.

Example 4.1. YAML configuration

For example, a configuration item documented as **`swarm.undertow.servers.KEY.default-host`** translates to the following YAML structure, substituting the **`KEY`** segment with the **`default`** identifier:

```
swarm:
  undertow:
    servers:
      default:
        default-host: <myhost>
```

4.7.2.2. Default Thorntail YAML Files

By default, Thorntail looks up permanent configuration in files with specific names to put on the classpath.

project-defaults.yml
If the original **`.war`** file with your application contains a file named **`project-defaults.yml`**, that file represents the defaults applied over the absolute defaults that Thorntail provides.

Other default file names
In addition to the **`project-defaults.yml`** file, you can provide specific configuration files using the **`-S <name>`** command-line option. The specified files are loaded, in the order you provided them, before **`project-defaults.yml`**. A name provided in the **`-S <name>`** argument specifies the **`project-<name>.yml`** file on your classpath.

Example 4.2. Specifying configuration files on the command line

Consider the following application execution:

```
$ java -jar myapp-thorntail.jar -Stesting -Scloud
```

The following YAML files are loaded, in this order. The first file containing a given configuration item takes precedence over others:

1. **`project-testing.yml`**
2. **`project-cloud.yml`**
3. **`project-defaults.yml`**

4.7.2.3. Non-default Thorntail YAML configuration files

In addition to default configuration files for your Thorntail-based application, you can specify YAML files outside of your application. Use the **`-s <path>`** command-line option to load the desired file.

Both the **`-s <path>`** and **`-S <name>`** command-line options can be used at the same time, but files specified using the **`-s <path>`** option take precedence over YAML files contained in your application.

Example 4.3. Specifying configuration files inside and outside of the application

Consider the following application execution:

```
$ java -jar myapp-thorntail.jar -s/home/app/openshift.yml -Scloud -
Stesting
```

The following YAML files are loaded, in this order:

1. **`/home/app/openshift.yml`**
2. **`project-cloud.yml`**
3. **`project-testing.yml`**
4. **`project-defaults.yml`**

The same order of preference is applied even if you invoke the application as follows:

```
$ java -jar myapp-thorntail.jar -Scloud -Stesting -
s/home/app/openshift.yml
```

Related information

- Section 4.7.2.2, “Default Thorntail YAML Files”

CHAPTER 5. PACKAGING YOUR APPLICATION

This sections contains information about packaging your Thorntail–based application for deployment and execution.

5.1. PACKAGING TYPES

When using Thorntail, there are the following ways to package your runtime and application, depending on how you intend to use and deploy it:

5.1.1. Uberjar

An *uberjar* is a single Java `.jar` file that includes everything you need to execute your application. This means both the runtime components you have selected—you can understand that as the app server—along with the application components (your `.war` file).

An uberjar is useful for many continuous integration and continuous deployment (CI/CD) pipeline styles, in which a single executable binary artifact is produced and moved through the testing, validation, and production environments in your organization.

The names of the uberjars that Thorntail produces include the name of your application and the `-thorntail.jar` suffix.

An uberjar can be executed like any executable JAR:

```
$ java -jar myapp-thorntail.jar
```

5.1.2. Hollow JAR

A *hollow JAR* is similar to an uberjar, but includes only the runtime components, and does not include your application code.

A hollow jar is suitable for deployment processes that involve Linux containers such as Docker. When using containers, place the runtime components in a container image lower in the image hierarchy—which means it changes less often—so that the higher layer which contains only your application code can be rebuilt more quickly.

The names of the hollow JARs that Thorntail produces include the name of your application, and the `-hollow-thorntail.jar` suffix. You must package the `.war` file of your application separately in order to benefit from the hollow JAR.

NOTE

Using hollow JARs has certain limitations:

- To enable Thorntail to autodetect a JDBC driver, you must add the JAR with the driver to the **`swarm.classpath`** system property, for example:

```
$ java -Dswarm.classpath=./h2-1.4.196.jar -jar my-hollow-thorntail.jar myApp.war
```

- YAML configuration files in your application are not automatically applied. You must specify them manually, for example:

```
$ java -jar my-hollow-thorntail.jar myApp.war -s ./project-defaults.yml
```

When executing the hollow JAR, provide the application **`.war`** file as an argument to the Java binary:

```
$ java -jar myapp-hollow-thorntail.jar myapp.war
```

5.1.2.1. Pre-Built Hollow JARs

Thorntail ships the following pre-built hollow JARs:

web
: Functionality focused on web technologies

microprofile
: Functionality defined by all Eclipse MicroProfile specifications

The hollow JARs are available under the following coordinates:

```
<dependency>
    <groupId>io.thorntail.servers</groupId>
    <artifactId>[web|microprofile]</artifactId>
</dependency>
```

5.2. CREATING AN UBERJAR

One method of packaging an application for execution with Thorntail is as an *uberjar*.

Prerequisites

- A Maven-based application with a **`pom.xml`** file.

Procedure

1. Add the **`thorntail-maven-plugin`** to your **`pom.xml`** in a **`<plugin>`** block, with an **`<execution>`** specifying the **`package`** goal.

```
<plugins>
  <plugin>
    <groupId>io.thorntail</groupId>
```

```
    <artifactId>thorntail-maven-plugin</artifactId>
    <version>${version.thorntail}</version>
    <executions>
      <execution>
        <id>package</id>
        <goals>
          <goal>package</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
```

2. Perform a normal Maven build:

```
$ mvn package
```

3. Execute the resulting uberjar:

```
$ java -jar ./target/myapp-thorntail.jar
```

CHAPTER 6. TESTING

6.1. TESTING IN A CONTAINER

Using Arquillian, you have the capability of injecting unit tests into a running application. This allows you to verify your application is behaving correctly. There is an adapter for Thorntail that makes Arquillian-based testing work well with Thorntail–based applications.

Prerequisites

- A Maven-based application with a **pom.xml** file.

Procedure

1. Include the Thorntail BOM as described in Section 4.5, “Using a BOM”:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.thorntail</groupId>
      <artifactId>bom</artifactId>
      <version>${version.thorntail}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

2. Reference the **io.thorntail:arquillian** artifact in your **pom.xml** file with the **<scope>** set to **test**:

```
<dependencies>
  <dependency>
    <groupId>io.thorntail</groupId>
    <artifactId>arquillian</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

3. Create your Application.
   Write your application as you normally would; use any default **project-defaults.yml** files you need to configure it.

```
swarm:
  datasources:
    data-sources:
      MyDS:
        driver-name: myh2
        connection-url: jdbc:h2:mem:test;DB_CLOSE_DELAY=-
1;DB_CLOSE_ON_EXIT=FALSE
        user-name: sa
        password: sa
    jdbc-drivers:
      myh2:
```

```
        driver-class-name: org.h2.Driver
        xa-datasource-name: org.h2.jdbcx.JdbcDataSource
        driver-module-name: com.h2database.h2
```

4. Create a test class.

NOTE

Creating an Arquillian test before Thorntail existed usually involved programatically creating **`Archive`** due to the fact that applications were larger, and the aim was to test a single component in isolation.

```
package org.wildfly.swarm.howto.incontainer;

public class InContainerTest {
}
```

5. Create a deployment.
In the context of microservices, the entire application represents one small microservice component.

   Use the **`@DefaultDeployment`** annotation to automatically create the deployment of the entire application. The **`@DefaultDeployment`** annotation defaults to creating a **`.war`** file, which is not applicable in this case because Undertow is not involved in this process.

   Apply the **`@DefaultDeployment`** annotation at the class level of a JUnit test, along with the **`@RunWith(Arquillian.class)`** annotation:

```
@RunWith(Arquillian.class)
@DefaultDeployment(type = DefaultDeployment.Type.JAR)
public class InContainerTest {
```

   Using the **`@DefaultDeployment`** annotation provided by Arquillian integration with Thorntail means you should *not* use the Arquillian **`@Deployment`** annotation on static methods that return an **`Archive`**.

   The **`@DefaultDeployment`** annotation inspects the package of the test:

```
package org.wildfly.swarm.howto.incontainer;
```

   From the package, it uses heuristics to include all of your other application classes in the same package or deeper in the Java packaging hierarchy.

   Even though using the **`@DefaultDeployment`** annotation allows you to write tests that only create a *default deployment* for sub-packages of your application, it also prevents you from placing tests in an unrelated package, for example:

```
package org.mycorp.myapp.test;
```

6. Write your test code.
Write an Arquillian-type of test as you normally would, including using Arquillian facilities to gain access to internal running components.

In the example below, Arquillian is used to inject the **`InitialContext`** of the running application into an instance member of the test case:

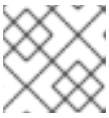
```
@ArquillianResource
InitialContext context;
```

That means the test method itself can use that **`InitialContext`** to ensure the Datasource you configured using **`project-defaults.yml`** is live and available:

```
@Test
public void testDataSourceIsBound() throws Exception {
    DataSource ds = (DataSource)
context.lookup("java:jboss/datasources/MyDS");
    assertNotNull( ds );
}
```

7. Run the tests.
   Because Arquillian provides an integration with JUnit, you can execute your test classes using Maven or your IDE:

```
$ mvn install
```



NOTE

In many IDEs, execute a test class by right-clicking it and selecting **Run**.

CHAPTER 7. DEBUGGING

This sections contains information about debugging your Thorntail–based application both in local and remote deployments.

7.1. REMOTE DEBUGGING

To remotely debug an application, you must first configure it to start in a debugging mode, and then attach a debugger to it.

7.1.1. Starting your application locally in debugging mode

One of the ways of debugging a Maven-based project is manually launching the application while specifying a debugging port, and subsequently connecting a remote debugger to that port. This method is applicable at least to the following deployments of the application:

- When launching the application manually using the **`mvn thorntail:run`** goal.
- When starting the application without waiting for it to exit using the **`mvn thorntail:start`** goal. This is useful especially when performing integration testing.
- When using the Arquillian adapter for Thorntail.

Prerequisites

- A Maven-based application

Procedure

1. In a console, navigate to the directory with your application.
2. Launch your application and specify the debug port using the **`-Dswarm.debug.port`** argument:

```
$ mvn thorntail:run -Dswarm.debug.port=$PORT_NUMBER
```

   Here, **`$PORT_NUMBER`** is an unused port number of your choice. Remember this number for the remote debugger configuration.

7.1.2. Starting an uberjar in debugging mode

If you chose to package your application as a Thorntail uberjar, debug it by executing it with the following parameters.

Prerequisites

- An uberjar with your application

Procedure

1. In a console, navigate to the directory with the uberjar.

2. Execute the uberjar with the following parameters. Ensure that all the parameters are specified before the name of the uberjar on the line.

```
$ java -
agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=$PORT_N
UMBER -jar $UBERJAR_FILENAME
```

$PORT_NUMBER is an unused port number of your choice. Remember this number for the remote debugger configuration.

If you want the JVM to pause and wait for remote debugger connection before it starts the application, change **suspend** to **y**.

Additional resources

- Section 5.2, “Creating an uberjar”

7.1.3. Starting your application on OpenShift in debugging mode

To debug your Thorntail-based application on OpenShift remotely, you must set the **JAVA_DEBUG** environment variable inside the container to **true** and configure port forwarding so that you can connect to your application from a remote debugger.

Prerequisites

- Your application running on OpenShift.
- The **oc** binary installed on your machine.
- The ability to execute the **oc port-forward** command in your target OpenShift environment.

Procedure

1. Using the **oc** command, list the available deployment configurations:

```
$ oc get dc
```

2. Set the **JAVA_DEBUG** environment variable in the deployment configuration of your application to **true**, which configures the JVM to open the port number **5005** for debugging. For example:

```
$ oc set env dc/MY_APP_NAME JAVA_DEBUG=true
```

3. Redeploy the application if it is not set to redeploy automatically on configuration change. For example:

```
$ oc rollout latest dc/MY_APP_NAME
```

4. Configure port forwarding from your local machine to the application pod:

   a. List the currently running pods and find one containing your application:

```
$ oc get pod
NAME                                READY     STATUS      RESTARTS
```

```
AGE
MY_APP_NAME-3-1xrsp          0/1       Running     0          6s
...
```

b. Configure port forwarding:

```
$ oc port-forward MY_APP_NAME-3-1xrsp $LOCAL_PORT_NUMBER:5005
```

Here, **$LOCAL_PORT_NUMBER** is an unused port number of your choice on your local machine. Remember this number for the remote debugger configuration.

5. When you are done debugging, unset the **JAVA_DEBUG** environment variable in your application pod. For example:

```
$ oc set env dc/MY_APP_NAME JAVA_DEBUG-
```

Additional resources

You can also set the **JAVA_DEBUG_PORT** environment variable if you want to change the debug port from the default, which is **5005**.

7.1.4. Attaching a remote debugger to the application

When your application is configured for debugging, attach a remote debugger of your choice to it. In this guide, Red Hat JBoss Developer Studio is covered, but the procedure is similar when using other programs.

Prerequisites

- The application running either locally or on OpenShift, and configured for debugging.
- The port number that your application is listening on for debugging.
- Red Hat JBoss Developer Studio installed on your machine. You can download it from the Red Hat JBoss Developer Studio download page.

Procedure

1. Start Red Hat JBoss Developer Studio.
2. Create a new debug configuration for your application:
   a. Click **Run→Debug Configurations**.
   b. In the list of configurations, double-click **Remote Java application.** This creates a new remote debugging configuration.
   c. Enter a suitable name for the configuration in the **Name** field.
   d. Enter the path to the directory with your application into the **Project** field. You can use the **Browse…** button for convenience.
   e. Set the **Connection Type** field to *Standard (Socket Attach)* if it is not already.
   f. Set the **Port** field to the port number that your application is listening on for debugging.

g. Click **Apply.**

3. Start debugging by clicking the **Debug** button in the Debug Configurations window.
To quickly launch your debug configuration after the first time, click **Run→Debug History** and select the configuration from the list.

Additional resources

- Debug an OpenShift Java Application with JBoss Developer Studio on Red Hat Knowledgebase.
- A Debugging Java Applications On OpenShift and Kubernetes article on OpenShift Blog.

7.2. DEBUG LOGGING

7.2.1. Local debug logging

To enable debug logging locally, see the Section 4.6.1, “Enabling logging” section and use the **`DEBUG`** log level.

If you want to enable debug logging permanently, add the following configuration to the **`src/main/resources/project-defaults.yml`** file in your application:

Debug logging YAML configuration

```
swarm:
  logging: DEBUG
```

7.2.2. Accessing debug logs on OpenShift

Start your application and interact with it to see the debugging statements in OpenShift.

Prerequisites

- A Maven-based application with debug logging enabled.
- The **oc** CLI client installed and authenticated.

Procedure

1. Deploy your application to OpenShift:

```
$ mvn clean fabric8:deploy -Popenshift
```

2. View the logs:
    1. Get the name of the pod with your application:

```
$ oc get pods
```

    2. Start watching the log output:

```
$ oc logs -f pod/MY_APP_NAME-2-aaaaa
```

Keep the terminal window displaying the log output open so that you can watch the log output.

3. Interact with your application:
   For example, if you had debug logging in the REST API Level 0 booster to log the **`message`** variable in the **`/api/greeting`** method:

   1. Get the route of your application:

      ```
      $ oc get routes
      ```

   2. Make an HTTP request on the **`/api/greeting`** endpoint of your application:

      ```
      $ curl $APPLICATION_ROUTE/api/greeting?name=Sarah
      ```

4. Return to the window with your pod logs and inspect debug logging messages in the logs.

   ```
   ...
   2018-02-11 11:12:31,158 INFO  [io.openshift.MY_APP_NAME] (default
   task-18) Hello, Sarah!
   ...
   ```

5. To disable debug logging, remove the **`logging`** key from the **`project-defaults.yml`** file and redeploy the appliation.

Additional resources

- Section 4.6, “Logging”

CHAPTER 8. MONITORING

This section contains information about monitoring your Thorntail–based application running on OpenShift.

8.1. ACCESSING JVM METRICS FOR YOUR APPLICATION ON OPENSHIFT

8.1.1. Accessing JVM metrics using Jolokia on OpenShift

Jolokia is a built-in lightweight solution for accessing JMX (Java Management Extension) metrics over HTTP on OpenShift. Jolokia allows you to access CPU, storage, and memory usage data collected by JMX over an HTTP bridge. Jolokia uses a REST interface and JSON-formatted message payloads. It is suitable for monitoring cloud applications thanks to its comparably high speed and low resource requirements.

For Java-based applications, the OpenShift Web console provides the integrated *hawt.io* console that collects and displays all relevant metrics output by the JVM running your application.

Prerequistes

- the **oc** client authenticated
- a Java-based application container running in a project on OpenShift
- latest JDK 1.8.0 image

Procedure

1. List the deployment configurations of the pods inside your project and select the one that corresponds to your application.

```
oc get dc
```

```
NAME          REVISION   DESIRED   CURRENT   TRIGGERED BY
MY_APP_NAME   2          1         1         config,image(my-app:6)
...
```

2. Open the YAML deployment template of the pod running your application for editing.

```
oc edit dc/MY_APP_NAME
```

3. Add the following entry to the **`ports`** section of the template and save your changes:

```
...
spec:
  ...
  ports:
  - containerPort: 8778
    name: jolokia
    protocol: TCP
  ...
...
```

4. Redeploy the pod running your application.

```
oc rollout latest dc/MY_APP_NAME
```

   The pod is redeployed with the updated deployment configuration and exposes the port **8778**.

5. Log into the OpenShift Web console.

6. In the sidebar, navigate to *Applications* > *Pods*, and click on the name of the pod running your application.

7. In the pod details screen, click *Open Java Console* to access the hawt.io console.

Additional resources

- hawt.io documentation

APPENDIX A. THE SOURCE-TO-IMAGE (S2I) BUILD PROCESS

Source-to-Image (S2I) is a build tool for generating reproducible Docker-formatted container images from online SCM repositories with application sources. With S2I builds, you can easily deliver the latest version of your application into production with shorter build times, decreased resource and network usage, improved security, and a number of other advantages. OpenShift supports multiple build strategies and input sources.

For more information, see the Source-to-Image (S2I) Build chapter of the OpenShift Container Platform documentation.

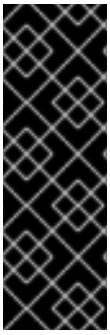
You must provide three elements to the S2I process to assemble the final container image:

- The application sources hosted in an online SCM repository, such as GitHub.
- The S2I Builder image, which serves as the foundation for the assembled image and provides the ecosystem in which your application is running.
- Optionally, you can also provide environment variables and parameters that are used by S2I scripts.

The process injects your application source and dependencies into the Builder image according to instructions specified in the S2I script, and generates a Docker-formatted container image that runs the assembled application. For more information, check the S2I build requirements, build options and how builds work sections of the OpenShift Container Platform documentation.

APPENDIX B. UPDATING THE DEPLOYMENT CONFIGURATION OF A BOOSTER

The deployment configuration for a booster contains information related to deploying and running the booster in OpenShift, such as route information or readiness probe location. The deployment configuration of a booster is stored in a set of YAML files. For boosters that use the Fabric8 Maven Plugin, the YAML files are located in the **`src/main/fabric8/`** directory. For boosters using Nodeshift, the YAML files are located in the **`.nodeshift`** directory.



IMPORTANT

The deployment configuration files used by the Fabric8 Maven Plugin and Nodeshift do not have to be full OpenShift resource definitions. Both Fabric8 Maven Plugin and Nodeshift can take the deployment configuration files and add some missing information to create a full OpenShift resource definition. The resource definitions generated by the Fabric8 Maven Plugin are available in the **`target/classes/META-INF/fabric8/`** directory. The resource definitions generated by Nodeshift are available in the **`tmp/nodeshift/resource/`** directory.

Prerequisites

- An existing booster project.
- The **oc** CLI client installed.

Procedure

1. Edit an existing YAML file or create an additional YAML file with your configuration update.
   - For example, if your booster already has a YAML file with a **`readinessProbe`** configured, you could change the **`path`** value to a different available path to check for readiness:

```
spec:
  template:
    spec:
      containers:
        readinessProbe:
          httpGet:
            path: /path/to/probe
            port: 8080
            scheme: HTTP
...
```

   - If a **`readinessProbe`** is not configured in an existing YAML file, you can also create a new YAML file in the same directory with the **`readinessProbe`** configuration.
2. Deploy the updated version of your booster using Maven or npm.
3. Verify that your configuration updates show in the deployed version of your booster.

```
$ oc export all --as-template='my-template'

apiVersion: v1
kind: Template
```

```
metadata:
  creationTimestamp: null
  name: my-template
objects:
- apiVersion: v1
  kind: DeploymentConfig
  ...
  spec:
    ...
    template:
      ...
      spec:
        containers:
          ...
          livenessProbe:
            failureThreshold: 3
            httpGet:
              path: /path/to/different/probe
              port: 8080
              scheme: HTTP
            initialDelaySeconds: 60
            periodSeconds: 30
            successThreshold: 1
            timeoutSeconds: 1
          ...
```

Additional resources

If you updated the configuration of your application directly using the web-based console or the **oc** CLI client, export and add these changes to your YAML file. Use the **`oc export all`** command to show the configuration of your deployed application.

APPENDIX C. CONFIGURING A JENKINS FREESTYLE PROJECT TO DEPLOY YOUR APPLICATION WITH THE FABRIC8 MAVEN PLUGIN

Similar to using Maven and the Fabric8 Maven Plugin from your local host to deploy an application, you can configure Jenkins to use Maven and the Fabric8 Maven Plugin to deploy an application.

Prerequisites

- Access to an OpenShift cluster.
- The Jenkins container image running on same OpenShift cluster.
- A JDK and Maven installed and configured on your Jenkins server.
- An application configured to use Maven, the Fabric8 Maven Plugin, and the Red Hat base image in the **`pom.xml`**.

  Example `pom.xml`

  ```
  <properties>
    ...
    <fabric8.generator.from>registry.access.redhat.com/redhat-openjdk-
  18/openjdk18-openshift</fabric8.generator.from>
  </properties>
  ```

- The source of the application available in GitHub.

Procedure

1. Create a new OpenShift project for your application:
   a. Open the OpenShift Web console and log in.
   b. Click *Create Project* to create a new OpenShift project.
   c. Enter the project information and click *Create*.
2. Ensure Jenkins has access to that project.
   For example, if you configured a service account for Jenkins, ensure that account has **`edit`** access to the project of your application.
3. Create a new freestyle Jenkins project on your Jenkins server:
   a. Click *New Item*.
   b. Enter a name, choose *Freestyle project*, and click *OK*.
   c. Under *Source Code Management*, choose *Git* and add the GitHub url of your application.
   d. Under *Build*, choose *Add build step* and select **`Invoke top-level Maven targets`**.
   e. Add the following to *Goals*:

```
clean fabric8:deploy -Popenshift -Dfabric8.namespace=MY_PROJECT
```

Substitute **`MY_PROJECT`** with the name of the OpenShift project for your application.

f. Click *Save*.

4. Click *Build Now* from the main page of the Jenkins project to verify your application builds and deploys to the OpenShift project for your application.
You can also verify that your application is deployed by opening the route in the OpenShift project of the application.

Next steps

- Consider adding GITSCM polling or using the **`Poll SCM`** build trigger. These options enable builds to run every time a new commit is pushed to the GitHub repository.
- Consider adding a build step that executes tests before deploying.

APPENDIX D. THORNTAIL FRACTIONS REFERENCE

For information about using the configuration properties provided in Thorntail fractions, see Section 4.7, “Configuring a Thorntail application”.

D.1. ARCHAIUS

WARNING

This fraction is deprecated.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>archaius</artifactId>
</dependency>
```

D.2. BEAN VALIDATION

Provides class-level constraint and validation according to JSR 303.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>bean-validation</artifactId>
</dependency>
```

D.3. CDI

Provides context and dependency-injection support according to JSR-299.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>cdi</artifactId>
</dependency>
```

Configuration

swarm.cdi.development-mode

Weld comes with a special mode for application development. When the development mode is enabled, certain built-in tools, which facilitate the development of CDI applications, are available. Setting this attribute to true activates the development mode.

swarm.cdi.non-portable-mode

If true then the non-portable mode is enabled. The non-portable mode is suggested by the specification to overcome problems with legacy applications that do not use CDI SPI properly and may be rejected by more strict validation in CDI 1.1.

swarm.cdi.require-bean-descriptor

If true then implicit bean archives without bean descriptor file (beans.xml) are ignored by Weld

swarm.cdi.thread-pool-size

The number of threads to be used by the Weld thread pool. The pool is shared across all CDI-enabled deployments and used primarily for parallel Weld bootstrap.

D.3.1. CDI Configuration

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>cdi-config</artifactId>
</dependency>
```

D.4. CONNECTOR

Primarily an internal fraction used to provide support for higher-level fractions such as JCA (JSR-322).

If you require JCA support, please see the JCA fraction documentation.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>connector</artifactId>
</dependency>
```

D.5. CONTAINER

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>container</artifactId>
</dependency>
```

D.6. DATASOURCES

Provides support for container-managed database connections.

D.6.1. Autodetectable drivers

If your application includes the appropriate vendor JDBC library in its normal dependencies, these drivers will be detected and installed by Thorntail without any additional effort.

The list of detectable drivers and their **`driver-name`** which may be used when defining a datasource is as follows:

Database	`driver-name`
MySQL	**`mysql`**
PostgreSQL	**`postgresql`**
H2	**`h2`**
EnterpriseDB	**`edb`**
IBM DB2	**`ibmdb2`**
Oracle DB	**`oracle`**
Microsoft SQLServer	**`sqlserver`**
Sybase	**`sybase`**
Teiid	**`teiid`**
MariaDB	**`mariadb`**
Derby	**`derby`**
Hive2	**`hive2`**
PrestoDB	**`prestodb`**

D.6.2. Example datasource definitions

D.6.2.1. MySQL

An example of a MySQL datasource configuration with connection information, basic security, and validation options:

```
swarm:
  datasources:
    data-sources:
      MyDS:
        driver-name: mysql
        connection-url: jdbc:mysql://localhost:3306/jbossdb
        user-name: admin
        password: admin
        valid-connection-checker-class-name:
org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker
        validate-on-match: true
```

```
        background-validation: false
        exception-sorter-class-name:
org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
```

D.6.2.2. PostgreSQL

An example of a PostgreSQL datasource configuration with connection information, basic security, and validation options:

```
swarm:
  datasources:
    data-sources:
      MyDS:
        driver-name: postgresql
        connection-url: jdbc:postgresql://localhost:5432/postgresdb
        user-name: admin
        password: admin
        valid-connection-checker-class-name:
org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionC
hecker
        validate-on-match: true
        background-validation: false
        exception-sorter-class-name:
org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter
```

D.6.2.3. Oracle

An example of an Oracle datasource configuration with connection information, basic security, and validation options:

```
swarm:
  datasources:
    data-sources:
      MyDS:
        driver-name: oracle
        connection-url: jdbc:oracle:thin:@localhost:1521:XE
        user-name: admin
        password: admin
        valid-connection-checker-class-name:
org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker
        validate-on-match: true
        background-validation: false
        stale-connection-checker-class-name:
org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker
        exception-sorter-class-name:
org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter
```

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>datasources</artifactId>
</dependency>
```

Configuration

swarm.datasources.data-sources.*KEY*.allocation-retry

The allocation retry element indicates the number of times that allocating a connection should be tried before throwing an exception

swarm.datasources.data-sources.*KEY*.allocation-retry-wait-millis

The allocation retry wait millis element specifies the amount of time, in milliseconds, to wait between retrying to allocate a connection

swarm.datasources.data-sources.*KEY*.allow-multiple-users

Specifies if multiple users will access the datasource through the getConnection(user, password) method and hence if the internal pool type should account for that

swarm.datasources.data-sources.*KEY*.authentication-context

The Elytron authentication context which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.

swarm.datasources.data-sources.*KEY*.background-validation

An element to specify that connections should be validated on a background thread versus being validated prior to use. Changing this value can be done only on disabled datasource, requires a server restart otherwise.

swarm.datasources.data-sources.*KEY*.background-validation-millis

The background-validation-millis element specifies the amount of time, in milliseconds, that background validation will run. Changing this value can be done only on disabled datasource, requires a server restart otherwise

swarm.datasources.data-sources.*KEY*.blocking-timeout-wait-millis

The blocking-timeout-millis element specifies the maximum time, in milliseconds, to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for locking a connection, and will never throw an exception if creating a new connection takes an inordinately long time

swarm.datasources.data-sources.*KEY*.capacity-decrementer-class

Class defining the policy for decrementing connections in the pool

swarm.datasources.data-sources.*KEY*.capacity-decrementer-properties

Properties to be injected in class defining the policy for decrementing connections in the pool

swarm.datasources.data-sources.*KEY*.capacity-incrementer-class

Class defining the policy for incrementing connections in the pool

swarm.datasources.data-sources.*KEY*.capacity-incrementer-properties

Properties to be injected in class defining the policy for incrementing connections in the pool

swarm.datasources.data-sources.*KEY*.check-valid-connection-sql

Specify an SQL statement to check validity of a pool connection. This may be called when managed connection is obtained from the pool

swarm.datasources.data-sources.*KEY*.connectable

Enable the use of CMR. This feature means that a local resource can reliably participate in an XA transaction.

swarm.datasources.data-sources.*KEY*.connection-listener-class

Speciefies class name extending org.jboss.jca.adapters.jdbc.spi.listener.ConnectionListener that provides a possible to listen for connection activation and passivation in order to perform actions before the connection is returned to the application or returned to the pool.

swarm.datasources.data-sources.*KEY*.connection-listener-property

Properties to be injected in class specidied in connection-listener-class

swarm.datasources.data-sources.*KEY*.connection-properties.*KEY*.value

Each connection-property specifies a string name/value pair with the property name coming from the name attribute and the value coming from the element content

swarm.datasources.data-sources.*KEY*.connection-url

The JDBC driver connection URL

swarm.datasources.data-sources.*KEY*.credential-reference

Credential (from Credential Store) to authenticate on data source

swarm.datasources.data-sources.*KEY*.datasource-class

The fully qualified name of the JDBC datasource class

swarm.datasources.data-sources.*KEY*.driver-class

The fully qualified name of the JDBC driver class

swarm.datasources.data-sources.*KEY*.driver-name

Defines the JDBC driver the datasource should use. It is a symbolic name matching the the name of installed driver. In case the driver is deployed as jar, the name is the name of deployment unit

swarm.datasources.data-sources.*KEY*.elytron-enabled

Enables Elytron security for handling authentication of connections. The Elytron authentication-context to be used will be current context if no context is specified (see authentication-context).

swarm.datasources.data-sources.*KEY*.enlistment-trace

Defines if WildFly/IronJacamar should record enlistment traces

swarm.datasources.data-sources.*KEY*.exception-sorter-class-name

An org.jboss.jca.adapters.jdbc.ExceptionSorter that provides an isExceptionFatal(SQLException) method to validate if an exception should broadcast an error

swarm.datasources.data-sources.*KEY*.exception-sorter-properties

The exception sorter properties

swarm.datasources.data-sources.*KEY*.flush-strategy

Specifies how the pool should be flush in case of an error.

swarm.datasources.data-sources.*KEY*.idle-timeout-minutes

The idle-timeout-minutes elements specifies the maximum time, in minutes, a connection may be idle before being closed. The actual maximum time depends also on the IdleRemover scan time, which is half of the smallest idle-timeout-minutes value of any pool. Changing this value can be done only on disabled datasource, requires a server restart otherwise.

swarm.datasources.data-sources.*KEY*.initial-pool-size

The initial-pool-size element indicates the initial number of connections a pool should hold.

swarm.datasources.data-sources.*KEY*.jndi-name

Specifies the JNDI name for the datasource

swarm.datasources.data-sources.*KEY*.jta

Enable JTA integration

swarm.datasources.data-sources.*KEY*.max-pool-size

The max-pool-size element specifies the maximum number of connections for a pool. No more connections will be created in each sub-pool

swarm.datasources.data-sources.*KEY*.mcp

Defines the ManagedConnectionPool implementation, f.ex. org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool

swarm.datasources.data-sources.*KEY*.min-pool-size

The min-pool-size element specifies the minimum number of connections for a pool

swarm.datasources.data-sources.*KEY*.new-connection-sql

Specifies an SQL statement to execute whenever a connection is added to the connection pool

swarm.datasources.data-sources.*KEY*.password

Specifies the password used when creating a new connection

swarm.datasources.data-sources.*KEY*.pool-fair

Defines if pool use should be fair

swarm.datasources.data-sources.*KEY*.pool-prefill

Should the pool be prefilled. Changing this value can be done only on disabled datasource, requires a server restart otherwise.

swarm.datasources.data-sources.*KEY*.pool-use-strict-min

Specifies if the min-pool-size should be considered strictly

swarm.datasources.data-sources.*KEY*.prepared-statements-cache-size

The number of prepared statements per connection in an LRU cache

swarm.datasources.data-sources.*KEY*.query-timeout

Any configured query timeout in seconds. If not provided no timeout will be set

swarm.datasources.data-sources.*KEY*.reauth-plugin-class-name

The fully qualified class name of the reauthentication plugin implementation

swarm.datasources.data-sources.*KEY*.reauth-plugin-properties

The properties for the reauthentication plugin

swarm.datasources.data-sources.*KEY*.security-domain

Specifies the PicketBox security domain which defines the PicketBox javax.security.auth.Subject that are used to distinguish connections in the pool

swarm.datasources.data-sources.*KEY*.set-tx-query-timeout

Whether to set the query timeout based on the time remaining until transaction timeout. Any configured query timeout will be used if there is no transaction

swarm.datasources.data-sources.*KEY*.share-prepared-statements

Whether to share prepared statements, i.e. whether asking for same statement twice without closing uses the same underlying prepared statement

swarm.datasources.data-sources.*KEY*.spy

Enable spying of SQL statements

swarm.datasources.data-sources.*KEY*.stale-connection-checker-class-name

An org.jboss.jca.adapters.jdbc.StaleConnectionChecker that provides an isStaleConnection(SQLException) method which if it returns true will wrap the exception in an org.jboss.jca.adapters.jdbc.StaleConnectionException

swarm.datasources.data-sources.*KEY*.stale-connection-checker-properties

The stale connection checker properties

swarm.datasources.data-sources.*KEY*.statistics-enabled

Define whether runtime statistics are enabled or not.

swarm.datasources.data-sources.*KEY*.track-statements

Whether to check for unclosed statements when a connection is returned to the pool, result sets are closed, a statement is closed or return to the prepared statement cache. Valid values are: "false" - do not track statements, "true" - track statements and result sets and warn when they are not closed,

"nowarn" - track statements but do not warn about them being unclosed

swarm.datasources.data-sources.*KEY*.tracking

Defines if IronJacamar should track connection handles across transaction boundaries

swarm.datasources.data-sources.*KEY*.transaction-isolation

Set the java.sql.Connection transaction isolation level. Valid values are: TRANSACTION_READ_UNCOMMITTED, TRANSACTION_READ_COMMITTED, TRANSACTION_REPEATABLE_READ, TRANSACTION_SERIALIZABLE and TRANSACTION_NONE. Different values are used to set customLevel using TransactionIsolation#customLevel

swarm.datasources.data-sources.*KEY*.url-delimiter

Specifies the delimiter for URLs in connection-url for HA datasources

swarm.datasources.data-sources.*KEY*.url-selector-strategy-class-name

A class that implements org.jboss.jca.adapters.jdbc.URLSelectorStrategy

swarm.datasources.data-sources.*KEY*.use-ccm

Enable the use of a cached connection manager

swarm.datasources.data-sources.*KEY*.use-fast-fail

Whether to fail a connection allocation on the first try if it is invalid (true) or keep trying until the pool is exhausted of all potential connections (false)

swarm.datasources.data-sources.*KEY*.use-java-context

Setting this to false will bind the datasource into global JNDI

swarm.datasources.data-sources.*KEY*.use-try-lock

Any configured timeout for internal locks on the resource adapter objects in seconds

swarm.datasources.data-sources.*KEY*.user-name

Specify the user name used when creating a new connection

swarm.datasources.data-sources.*KEY*.valid-connection-checker-class-name

An org.jboss.jca.adapters.jdbc.ValidConnectionChecker that provides an isValidConnection(Connection) method to validate a connection. If an exception is returned that means the connection is invalid. This overrides the check-valid-connection-sql element

swarm.datasources.data-sources.*KEY*.valid-connection-checker-properties

The valid connection checker properties

swarm.datasources.data-sources.*KEY*.validate-on-match

The validate-on-match element specifies if connection validation should be done when a connection factory attempts to match a managed connection. This is typically exclusive to the use of background validation

swarm.datasources.installed-drivers

List of JDBC drivers that have been installed in the runtime

swarm.datasources.jdbc-drivers.*KEY*.deployment-name

The name of the deployment unit from which the driver was loaded

swarm.datasources.jdbc-drivers.*KEY*.driver-class-name

The fully qualified class name of the java.sql.Driver implementation

swarm.datasources.jdbc-drivers.*KEY*.driver-datasource-class-name

The fully qualified class name of the javax.sql.DataSource implementation

swarm.datasources.jdbc-drivers.*KEY*.driver-major-version

The driver’s major version number

swarm.datasources.jdbc-drivers.*KEY*.driver-minor-version

The driver's minor version number

swarm.datasources.jdbc-drivers.*KEY*.driver-module-name

The name of the module from which the driver was loaded, if it was loaded from the module path

swarm.datasources.jdbc-drivers.*KEY*.driver-name

Defines the JDBC driver the datasource should use. It is a symbolic name matching the the name of installed driver. In case the driver is deployed as jar, the name is the name of deployment unit

swarm.datasources.jdbc-drivers.*KEY*.driver-xa-datasource-class-name

The fully qualified class name of the javax.sql.XADataSource implementation

swarm.datasources.jdbc-drivers.*KEY*.jdbc-compliant

Whether or not the driver is JDBC compliant

swarm.datasources.jdbc-drivers.*KEY*.module-slot

The slot of the module from which the driver was loaded, if it was loaded from the module path

swarm.datasources.jdbc-drivers.*KEY*.profile

Domain Profile in which driver is defined. Null in case of standalone server

swarm.datasources.jdbc-drivers.*KEY*.xa-datasource-class

XA datasource class

swarm.datasources.xa-data-sources.*KEY*.allocation-retry

The allocation retry element indicates the number of times that allocating a connection should be tried before throwing an exception

swarm.datasources.xa-data-sources.*KEY*.allocation-retry-wait-millis

The allocation retry wait millis element specifies the amount of time, in milliseconds, to wait between retrying to allocate a connection

swarm.datasources.xa-data-sources.*KEY*.allow-multiple-users

Specifies if multiple users will access the datasource through the getConnection(user, password) method and hence if the internal pool type should account for that

swarm.datasources.xa-data-sources.*KEY*.authentication-context

The Elytron authentication context which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.

swarm.datasources.xa-data-sources.*KEY*.background-validation

An element to specify that connections should be validated on a background thread versus being validated prior to use.

swarm.datasources.xa-data-sources.*KEY*.background-validation-millis

The background-validation-millis element specifies the amount of time, in milliseconds, that background validation will run.

swarm.datasources.xa-data-sources.*KEY*.blocking-timeout-wait-millis

The blocking-timeout-millis element specifies the maximum time, in milliseconds, to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for locking a connection, and will never throw an exception if creating a new connection takes an inordinately long time

swarm.datasources.xa-data-sources.*KEY*.capacity-decrementer-class

Class defining the policy for decrementing connections in the pool

swarm.datasources.xa-data-sources.*KEY*.capacity-decrementer-properties

Properties to inject in class defining the policy for decrementing connections in the pool

swarm.datasources.xa-data-sources.*KEY*.capacity-incrementer-class

Class defining the policy for incrementing connections in the pool

swarm.datasources.xa-data-sources.*KEY*.capacity-incrementer-properties

Properties to inject in class defining the policy for incrementing connections in the pool

swarm.datasources.xa-data-sources.*KEY*.check-valid-connection-sql

Specify an SQL statement to check validity of a pool connection. This may be called when managed connection is obtained from the pool

swarm.datasources.xa-data-sources.*KEY*.connectable

Enable the use of CMR for this datasource. This feature means that a local resource can reliably participate in an XA transaction.

swarm.datasources.xa-data-sources.*KEY*.connection-listener-class

Speciefies class name extending org.jboss.jca.adapters.jdbc.spi.listener.ConnectionListener that provides a possible to listen for connection activation and passivation in order to perform actions before the connection is returned to the application or returned to the pool.

swarm.datasources.xa-data-sources.*KEY*.connection-listener-property

Properties to be injected in class specified in connection-listener-class

swarm.datasources.xa-data-sources.*KEY*.credential-reference

Credential (from Credential Store) to authenticate on data source

swarm.datasources.xa-data-sources.*KEY*.driver-name

Defines the JDBC driver the datasource should use. It is a symbolic name matching the the name of installed driver. In case the driver is deployed as jar, the name is the name of deployment unit

swarm.datasources.xa-data-sources.*KEY*.elytron-enabled

Enables Elytron security for handling authentication of connections for recovery. The Elytron authentication-context to be used will be current context if no context is specified (see authentication-context).

swarm.datasources.xa-data-sources.*KEY*.enlistment-trace

Defines if WildFly/IronJacamar should record enlistment traces

swarm.datasources.xa-data-sources.*KEY*.exception-sorter-class-name

An org.jboss.jca.adapters.jdbc.ExceptionSorter that provides an isExceptionFatal(SQLException) method to validate if an exception should broadcast an error

swarm.datasources.xa-data-sources.*KEY*.exception-sorter-properties

The exception sorter properties

swarm.datasources.xa-data-sources.*KEY*.flush-strategy

Specifies how the pool should be flush in case of an error.

swarm.datasources.xa-data-sources.*KEY*.idle-timeout-minutes

The idle-timeout-minutes elements specifies the maximum time, in minutes, a connection may be idle before being closed. The actual maximum time depends also on the IdleRemover scan time, which is half of the smallest idle-timeout-minutes value of any pool. Changing this value can be done only on disabled datasource, requires a server restart otherwise.

swarm.datasources.xa-data-sources.*KEY*.initial-pool-size

The initial-pool-size element indicates the initial number of connections a pool should hold.

swarm.datasources.xa-data-sources.*KEY*.interleaving

An element to enable interleaving for XA connections

swarm.datasources.xa-data-sources.*KEY*.jndi-name

Specifies the JNDI name for the datasource

swarm.datasources.xa-data-sources.*KEY*.max-pool-size

The max-pool-size element specifies the maximum number of connections for a pool. No more connections will be created in each sub-pool

swarm.datasources.xa-data-sources.*KEY*.mcp

Defines the ManagedConnectionPool implementation, f.ex. org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool

swarm.datasources.xa-data-sources.*KEY*.min-pool-size

The min-pool-size element specifies the minimum number of connections for a pool

swarm.datasources.xa-data-sources.*KEY*.new-connection-sql

Specifies an SQL statement to execute whenever a connection is added to the connection pool

swarm.datasources.xa-data-sources.*KEY*.no-recovery

Specifies if the connection pool should be excluded from recovery

swarm.datasources.xa-data-sources.*KEY*.no-tx-separate-pool

Oracle does not like XA connections getting used both inside and outside a JTA transaction. To workaround the problem you can create separate sub-pools for the different contexts

swarm.datasources.xa-data-sources.*KEY*.pad-xid

Should the Xid be padded

swarm.datasources.xa-data-sources.*KEY*.password

Specifies the password used when creating a new connection

swarm.datasources.xa-data-sources.*KEY*.pool-fair

Defines if pool use should be fair

swarm.datasources.xa-data-sources.*KEY*.pool-prefill

Should the pool be prefilled. Changing this value can be done only on disabled datasource, requires a server restart otherwise.

swarm.datasources.xa-data-sources.*KEY*.pool-use-strict-min

Specifies if the min-pool-size should be considered strictly

swarm.datasources.xa-data-sources.*KEY*.prepared-statements-cache-size

The number of prepared statements per connection in an LRU cache

swarm.datasources.xa-data-sources.*KEY*.query-timeout

Any configured query timeout in seconds. If not provided no timeout will be set

swarm.datasources.xa-data-sources.*KEY*.reauth-plugin-class-name

The fully qualified class name of the reauthentication plugin implementation

swarm.datasources.xa-data-sources.*KEY*.reauth-plugin-properties

The properties for the reauthentication plugin

swarm.datasources.xa-data-sources.*KEY*.recovery-authentication-context

The Elytron authentication context which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.

swarm.datasources.xa-data-sources.*KEY*.recovery-credential-reference

Credential (from Credential Store) to authenticate on data source

swarm.datasources.xa-data-sources.*KEY*.recovery-elytron-enabled

Enables Elytron security for handling authentication of connections for recovery. The Elytron authentication-context to be used will be current context if no context is specified (see authentication-context).

swarm.datasources.xa-data-sources.*KEY*.recovery-password

The password used for recovery

swarm.datasources.xa-data-sources.*KEY*.recovery-plugin-class-name

The fully qualified class name of the recovery plugin implementation

swarm.datasources.xa-data-sources.*KEY*.recovery-plugin-properties

The properties for the recovery plugin

swarm.datasources.xa-data-sources.*KEY*.recovery-security-domain

The security domain used for recovery

swarm.datasources.xa-data-sources.*KEY*.recovery-username

The user name used for recovery

swarm.datasources.xa-data-sources.*KEY*.same-rm-override

The is-same-rm-override element allows one to unconditionally set whether the javax.transaction.xa.XAResource.isSameRM(XAResource) returns true or false

swarm.datasources.xa-data-sources.*KEY*.security-domain

Specifies the PicketBox security domain which defines the javax.security.auth.Subject that are used to distinguish connections in the pool

swarm.datasources.xa-data-sources.*KEY*.set-tx-query-timeout

Whether to set the query timeout based on the time remaining until transaction timeout. Any configured query timeout will be used if there is no transaction

swarm.datasources.xa-data-sources.*KEY*.share-prepared-statements

Whether to share prepared statements, i.e. whether asking for same statement twice without closing uses the same underlying prepared statement

swarm.datasources.xa-data-sources.*KEY*.spy

Enable spying of SQL statements

swarm.datasources.xa-data-sources.*KEY*.stale-connection-checker-class-name

An org.jboss.jca.adapters.jdbc.StaleConnectionChecker that provides an isStaleConnection(SQLException) method which if it returns true will wrap the exception in an org.jboss.jca.adapters.jdbc.StaleConnectionException

swarm.datasources.xa-data-sources.*KEY*.stale-connection-checker-properties

The stale connection checker properties

swarm.datasources.xa-data-sources.*KEY*.statistics-enabled

Define whether runtime statistics are enabled or not.

swarm.datasources.xa-data-sources.*KEY*.track-statements

Whether to check for unclosed statements when a connection is returned to the pool, result sets are closed, a statement is closed or return to the prepared statement cache. Valid values are: "false" - do not track statements, "true" - track statements and result sets and warn when they are not closed, "nowarn" - track statements but do not warn about them being unclosed

swarm.datasources.xa-data-sources.*KEY*.tracking

Defines if IronJacamar should track connection handles across transaction boundaries

swarm.datasources.xa-data-sources.*KEY*.transaction-isolation

Set the java.sql.Connection transaction isolation level. Valid values are: TRANSACTION_READ_UNCOMMITTED, TRANSACTION_READ_COMMITTED,

TRANSACTION_REPEATABLE_READ, TRANSACTION_SERIALIZABLE and TRANSACTION_NONE. Different values are used to set customLevel using TransactionIsolation#customLevel.

swarm.datasources.xa-data-sources.*KEY*.url-delimiter

Specifies the delimiter for URLs in connection-url for HA datasources

swarm.datasources.xa-data-sources.*KEY*.url-property

Specifies the property for the URL property in the xa-datasource-property values

swarm.datasources.xa-data-sources.*KEY*.url-selector-strategy-class-name

A class that implements org.jboss.jca.adapters.jdbc.URLSelectorStrategy

swarm.datasources.xa-data-sources.*KEY*.use-ccm

Enable the use of a cached connection manager

swarm.datasources.xa-data-sources.*KEY*.use-fast-fail

Whether to fail a connection allocation on the first try if it is invalid (true) or keep trying until the pool is exhausted of all potential connections (false)

swarm.datasources.xa-data-sources.*KEY*.use-java-context

Setting this to false will bind the datasource into global JNDI

swarm.datasources.xa-data-sources.*KEY*.use-try-lock

Any configured timeout for internal locks on the resource adapter objects in seconds

swarm.datasources.xa-data-sources.*KEY*.user-name

Specify the user name used when creating a new connection

swarm.datasources.xa-data-sources.*KEY*.valid-connection-checker-class-name

An org.jboss.jca.adapters.jdbc.ValidConnectionChecker that provides an isValidConnection(Connection) method to validate a connection. If an exception is returned that means the connection is invalid. This overrides the check-valid-connection-sql element

swarm.datasources.xa-data-sources.*KEY*.valid-connection-checker-properties

The valid connection checker properties

swarm.datasources.xa-data-sources.*KEY*.validate-on-match

The validate-on-match element specifies if connection validation should be done when a connection factory attempts to match a managed connection. This is typically exclusive to the use of background validation

swarm.datasources.xa-data-sources.*KEY*.wrap-xa-resource

Should the XAResource instances be wrapped in an org.jboss.tm.XAResourceWrapper instance

swarm.datasources.xa-data-sources.*KEY*.xa-datasource-class

The fully qualified name of the javax.sql.XADataSource implementation

swarm.datasources.xa-data-sources.*KEY*.xa-datasource-properties.*KEY*.value

Specifies a property value to assign to the XADataSource implementation class. Each property is identified by the name attribute and the property value is given by the xa-datasource-property element content. The property is mapped onto the XADataSource implementation by looking for a JavaBeans style getter method for the property name. If found, the value of the property is set using the JavaBeans setter with the element text translated to the true property type using the java.beans.PropertyEditor

swarm.datasources.xa-data-sources.*KEY*.xa-resource-timeout

The value is passed to XAResource.setTransactionTimeout(), in seconds. Default is zero

swarm.ds.connection.url

Default datasource connection URL

swarm.ds.name

Name of the default datasource

swarm.ds.password

Defatul datasource connection password

swarm.ds.username

Default datasource connection user name

swarm.jdbc.driver

Defatul datasource JDBC driver name

D.7. EE

An internal fraction used to support other higher-level fractions.

The EE fraction does *not* imply the totality of Java EE support.

If you require specific Java EE technologies, address them individually, for example **`jaxrs`**, **`cdi`**, **`datasources`**, or **`ejb`**.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>ee</artifactId>
</dependency>
```

Configuration

swarm.ee.annotation-property-replacement

Flag indicating whether Java EE annotations will have property replacements applied

swarm.ee.context-services.*KEY*.jndi-name

The JNDI Name to lookup the context service.

swarm.ee.context-services.*KEY*.use-transaction-setup-provider

Flag which indicates if the transaction setup provider should be used

swarm.ee.default-bindings-service.context-service

The JNDI name where the default EE Context Service can be found

swarm.ee.default-bindings-service.datasource

The JNDI name where the default EE Datasource can be found

swarm.ee.default-bindings-service.jms-connection-factory

The JNDI name where the default EE JMS Connection Factory can be found

swarm.ee.default-bindings-service.managed-executor-service

The JNDI name where the default EE Managed Executor Service can be found

swarm.ee.default-bindings-service.managed-scheduled-executor-service

The JNDI name where the default EE Managed Scheduled Executor Service can be found

swarm.ee.default-bindings-service.managed-thread-factory

The JNDI name where the default EE Managed Thread Factory can be found

swarm.ee.ear-subdeployments-isolated

Flag indicating whether each of the subdeployments within a .ear can access classes belonging to another subdeployment within the same .ear. A value of false means the subdeployments can see classes belonging to other subdeployments within the .ear.

swarm.ee.global-modules

A list of modules that should be made available to all deployments.

swarm.ee.jboss-descriptor-property-replacement

Flag indicating whether JBoss specific deployment descriptors will have property replacements applied

swarm.ee.managed-executor-services.*KEY*.context-service

The name of the context service to be used by the executor.

swarm.ee.managed-executor-services.*KEY*.core-threads

The minimum number of threads to be used by the executor. If left undefined the default core-size is calculated based on the number of processors. A value of zero is not advised and in some cases invalid. See the queue-length attribute for details on how this value is used to determine the queuing strategy.

swarm.ee.managed-executor-services.*KEY*.hung-task-threshold

The runtime, in milliseconds, for tasks to be considered hung by the managed executor service. If value is 0 tasks are never considered hung.

swarm.ee.managed-executor-services.*KEY*.jndi-name

The JNDI Name to lookup the managed executor service.

swarm.ee.managed-executor-services.*KEY*.keepalive-time

When the number of threads is greater than the core, this is the maximum time, in milliseconds, that excess idle threads will wait for new tasks before terminating.

swarm.ee.managed-executor-services.*KEY*.long-running-tasks

Flag which hints the duration of tasks executed by the executor.

swarm.ee.managed-executor-services.*KEY*.max-threads

The maximum number of threads to be used by the executor. If left undefined the value from core-size will be used. This value is ignored if an unbounded queue is used (only core-threads will be used in that case).

swarm.ee.managed-executor-services.*KEY*.queue-length

The executors task queue capacity. A length of 0 means direct hand-off and possible rejection will occur. An undefined length (the default), or Integer.MAX_VALUE, indicates that an unbounded queue should be used. All other values specify an exact queue size. If an unbounded queue or direct hand-off is used, a core-threads value greater than zero is required.

swarm.ee.managed-executor-services.*KEY*.reject-policy

The policy to be applied to aborted tasks.

swarm.ee.managed-executor-services.*KEY*.thread-factory

The name of the thread factory to be used by the executor.

swarm.ee.managed-scheduled-executor-services.*KEY*.context-service

The name of the context service to be used by the scheduled executor.

swarm.ee.managed-scheduled-executor-services.*KEY*.core-threads

The minimum number of threads to be used by the scheduled executor.

swarm.ee.managed-scheduled-executor-services.*KEY*.hung-task-threshold

The runtime, in milliseconds, for tasks to be considered hung by the scheduled executor. If 0 tasks are never considered hung.

swarm.ee.managed-scheduled-executor-services.*KEY*.jndi-name

The JNDI Name to lookup the managed scheduled executor service.

swarm.ee.managed-scheduled-executor-services.*KEY*.keepalive-time

When the number of threads is greater than the core, this is the maximum time, in milliseconds, that excess idle threads will wait for new tasks before terminating.

swarm.ee.managed-scheduled-executor-services.*KEY*.long-running-tasks

Flag which hints the duration of tasks executed by the scheduled executor.

swarm.ee.managed-scheduled-executor-services.*KEY*.reject-policy

The policy to be applied to aborted tasks.

swarm.ee.managed-scheduled-executor-services.*KEY*.thread-factory

The name of the thread factory to be used by the scheduled executor.

swarm.ee.managed-thread-factories.*KEY*.context-service

The name of the context service to be used by the managed thread factory

swarm.ee.managed-thread-factories.*KEY*.jndi-name

The JNDI Name to lookup the managed thread factory.

swarm.ee.managed-thread-factories.*KEY*.priority

The priority applied to threads created by the factory

swarm.ee.spec-descriptor-property-replacement

Flag indicating whether descriptors defined by the Java EE specification will have property replacements applied

D.8. EJB

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>ejb</artifactId>
</dependency>
```

Configuration

swarm.ejb3.allow-ejb-name-regex

If this is true then regular expressions can be used in interceptor bindings to allow interceptors to be mapped to all beans that match the regular expression

swarm.ejb3.application-security-domains.*KEY*.enable-jacc

Enable authorization using JACC

swarm.ejb3.application-security-domains.*KEY*.referencing-deployments

The deployments currently referencing this mapping

swarm.ejb3.application-security-domains.*KEY*.security-domain

The Elytron security domain to be used by deployments that reference the mapped security domain

swarm.ejb3.async-service.thread-pool-name

The name of the thread pool which handles asynchronous invocations

swarm.ejb3.caches.*KEY*.aliases

The aliases by which this cache may also be referenced

swarm.ejb3.caches.*KEY*.passivation-store

The passivation store used by this cache

swarm.ejb3.cluster-passivation-stores.*KEY*.bean-cache

The name of the cache used to store bean instances.

swarm.ejb3.cluster-passivation-stores.*KEY*.cache-container

The name of the cache container used for the bean and client-mappings caches

swarm.ejb3.cluster-passivation-stores.*KEY*.idle-timeout

The timeout in units specified by idle-timeout-unit, after which a bean will passivate

swarm.ejb3.cluster-passivation-stores.*KEY*.max-size

The maximum number of beans this cache should store before forcing old beans to passivate

swarm.ejb3.default-clustered-sfsb-cache

Name of the default stateful bean cache, which will be applicable to all clustered stateful EJBs, unless overridden at the deployment or bean level

swarm.ejb3.default-distinct-name

The default distinct name that is applied to every EJB deployed on this server

swarm.ejb3.default-entity-bean-instance-pool

Name of the default entity bean instance pool, which will be applicable to all entity beans, unless overridden at the deployment or bean level

swarm.ejb3.default-entity-bean-optimistic-locking

If set to true entity beans will use optimistic locking by default

swarm.ejb3.default-mdb-instance-pool

Name of the default MDB instance pool, which will be applicable to all MDBs, unless overridden at the deployment or bean level

swarm.ejb3.default-missing-method-permissions-deny-access

If this is set to true then methods on an EJB with a security domain specified or with other methods with security metadata will have an implicit @DenyAll unless other security metadata is present

swarm.ejb3.default-resource-adapter-name

Name of the default resource adapter name that will be used by MDBs, unless overridden at the deployment or bean level

swarm.ejb3.default-security-domain

The default security domain that will be used for EJBs if the bean doesn't explicitly specify one

swarm.ejb3.default-sfsb-cache

Name of the default stateful bean cache, which will be applicable to all stateful EJBs, unless overridden at the deployment or bean level

swarm.ejb3.default-sfsb-passivation-disabled-cache

Name of the default stateful bean cache, which will be applicable to all stateful EJBs which have passivation disabled. Each deployment or EJB can optionally override this cache name.

swarm.ejb3.default-singleton-bean-access-timeout

The default access timeout for singleton beans

swarm.ejb3.default-slsb-instance-pool

Name of the default stateless bean instance pool, which will be applicable to all stateless EJBs, unless overridden at the deployment or bean level

swarm.ejb3.default-stateful-bean-access-timeout

The default access timeout for stateful beans

swarm.ejb3.disable-default-ejb-permissions

This deprecated attribute has no effect and will be removed in a future release; it may never be set to a "false" value

swarm.ejb3.enable-graceful-txn-shutdown

Enabling txn graceful shutdown will make the server wait for active EJB-related transactions to complete before suspending. For that reason, if the server is running on a cluster, the suspending cluster node may receive ejb requests until all active transactions are complete. To avoid this behavior, omit this tag.

swarm.ejb3.enable-statistics

If set to true, enable the collection of invocation statistics. Deprecated in favour of "statistics-enabled"

swarm.ejb3.file-passivation-stores.*KEY*.idle-timeout

The timeout in units specified by idle-timeout-unit, after which a bean will passivate

swarm.ejb3.file-passivation-stores.*KEY*.max-size

The maximum number of beans this cache should store before forcing old beans to passivate

swarm.ejb3.identity-service.outflow-security-domains

References to security domains to attempt to outflow any established identity to

swarm.ejb3.iiop-service.enable-by-default

If this is true EJB's will be exposed over IIOP by default, otherwise it needs to be explicitly enabled in the deployment descriptor

swarm.ejb3.iiop-service.use-qualified-name

If true EJB names will be bound into the naming service with the application and module name prepended to the name (e.g. myapp/mymodule/MyEjb)

swarm.ejb3.in-vm-remote-interface-invocation-pass-by-value

If set to false, the parameters to invocations on remote interface of an EJB, will be passed by reference. Else, the parameters will be passed by value.

swarm.ejb3.log-system-exceptions

If this is true then all EJB system (not application) exceptions will be logged. The EJB spec mandates this behaviour, however it is not recommended as it will often result in exceptions being logged twice (once by the EJB and once by the calling code)

swarm.ejb3.mdb-delivery-groups.*KEY*.active

Indicates if delivery for all MDBs belonging to this group is active

swarm.ejb3.passivation-stores.*KEY*.bean-cache

The name of the cache used to store bean instances.

swarm.ejb3.passivation-stores.*KEY*.cache-container

The name of the cache container used for the bean and client-mappings caches

swarm.ejb3.passivation-stores.*KEY*.max-size

The maximum number of beans this cache should store before forcing old beans to passivate

swarm.ejb3.remote-service.channel-creation-options.*KEY*.type

The type of the channel creation option

swarm.ejb3.remote-service.channel-creation-options.*KEY*.value

The value for the EJB remote channel creation option

swarm.ejb3.remote-service.cluster

The name of the clustered cache container which will be used to store/access the client-mappings of the EJB remoting connector's socket-binding on each node, in the cluster

swarm.ejb3.remote-service.connector-ref

The name of the connector on which the EJB3 remoting channel is registered

swarm.ejb3.remote-service.execute-in-worker

If this is true the EJB request will be executed in the IO subsystems worker, otherwise it will dispatch to the EJB thread pool

swarm.ejb3.remote-service.thread-pool-name

The name of the thread pool that handles remote invocations

swarm.ejb3.remoting-profiles.*KEY*.exclude-local-receiver

If set no local receiver is used in this profile

swarm.ejb3.remoting-profiles.*KEY*.local-receiver-pass-by-value

If set local receiver will pass ejb beans by value

swarm.ejb3.remoting-profiles.*KEY*.remoting-ejb-receivers.*KEY*.channel-creation-options.*KEY*.type

The type of the channel creation option

swarm.ejb3.remoting-profiles.*KEY*.remoting-ejb-receivers.*KEY*.channel-creation-options.*KEY*.value

The value for the EJB remote channel creation option

swarm.ejb3.remoting-profiles.*KEY*.remoting-ejb-receivers.*KEY*.connect-timeout

Remoting ejb receiver connect timeout

swarm.ejb3.remoting-profiles.*KEY*.remoting-ejb-receivers.*KEY*.outbound-connection-ref

Name of outbound connection that will be used by the ejb receiver

swarm.ejb3.remoting-profiles.*KEY*.static-ejb-discovery

Describes static discovery config for EJB's

swarm.ejb3.statistics-enabled

If set to true, enable the collection of invocation statistics.

swarm.ejb3.strict-max-bean-instance-pools.*KEY*.derive-size

Specifies if and what the max pool size should be derived from. An undefined value (or the deprecated value 'none' which is converted to undefined) indicates that the explicit value of max-pool-size should be used. A value of 'from-worker-pools' indicates that the max pool size should be derived from the size of the total threads for all worker pools configured on the system. A value of 'from-cpu-count' indicates that the max pool size should be derived from the total number of processors available on the system. Note that the computation isn't a 1:1 mapping, the values may or may not be augmented by other factors.

swarm.ejb3.strict-max-bean-instance-pools.*KEY*.max-pool-size

The maximum number of bean instances that the pool can hold at a given point in time

swarm.ejb3.strict-max-bean-instance-pools.*KEY*.timeout

The maximum amount of time to wait for a bean instance to be available from the pool

swarm.ejb3.strict-max-bean-instance-pools.*KEY*.timeout-unit

The instance acquisition timeout unit

swarm.ejb3.thread-pools.*KEY*.active-count

The approximate number of threads that are actively executing tasks.

swarm.ejb3.thread-pools.*KEY*.completed-task-count

The approximate total number of tasks that have completed execution.

swarm.ejb3.thread-pools.*KEY*.current-thread-count

The current number of threads in the pool.

swarm.ejb3.thread-pools.*KEY*.keepalive-time

Used to specify the amount of time that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

swarm.ejb3.thread-pools.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

swarm.ejb3.thread-pools.*KEY*.max-threads

The maximum thread pool size.

swarm.ejb3.thread-pools.*KEY*.name

The name of the thread pool.

swarm.ejb3.thread-pools.*KEY*.queue-size

The queue size.

swarm.ejb3.thread-pools.*KEY*.rejected-count

The number of tasks that have been rejected.

swarm.ejb3.thread-pools.*KEY*.task-count

The approximate total number of tasks that have ever been scheduled for execution.

swarm.ejb3.thread-pools.*KEY*.thread-factory

Specifies the name of a specific thread factory to use to create worker threads. If not defined an appropriate default thread factory will be used.

swarm.ejb3.timer-service.database-data-stores.*KEY*.allow-execution

If this node is allowed to execute timers. If this is false then the timers will be added to the database, and another node may execute them. Note that depending on your refresh interval if you add timers with a very short delay they will not be executed until another node refreshes.

swarm.ejb3.timer-service.database-data-stores.*KEY*.database

The type of database that is in use. SQL can be customised per database type.

swarm.ejb3.timer-service.database-data-stores.*KEY*.datasource-jndi-name

The datasource that is used to persist the timers

swarm.ejb3.timer-service.database-data-stores.*KEY*.partition

The partition name. This should be set to a different value for every node that is sharing a database to prevent the same timer being loaded by multiple noded.

swarm.ejb3.timer-service.database-data-stores.*KEY*.refresh-interval

Interval between refreshing the current timer set against the underlying database. A low value means timers get picked up more quickly, but increase load on the database.

swarm.ejb3.timer-service.default-data-store

The default data store used for persistent timers

swarm.ejb3.timer-service.file-data-stores.*KEY*.path

The directory to store persistent timer information in

swarm.ejb3.timer-service.file-data-stores.*KEY*.relative-to

The relative path that is used to resolve the timer data store location

swarm.ejb3.timer-service.thread-pool-name

The name of the thread pool used to run timer service invocations

D.9. ELYTRON

Elytron can generate the audit log to the same directory where the Thorntail application is executed. Include the following section in the **`project-defaults.yml`** file in your application:

```
swarm:
  elytron:
    file-audit-logs:
      local-audit:
        path: audit.log
```

In some environments, for example cloud, you might have to relocate the audit file to a globally writable directory, for example:

```
swarm:
  elytron:
    file-audit-logs:
      local-audit:
        path: /tmp/audit.log
```

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>elytron</artifactId>
</dependency>
```

Configuration

swarm.elytron.add-prefix-role-mappers.*KEY*.prefix

The prefix to add to each role.

swarm.elytron.add-suffix-role-mappers.*KEY*.suffix

The suffix to add to each role.

swarm.elytron.aggregate-http-server-mechanism-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this factory instance.

swarm.elytron.aggregate-http-server-mechanism-factories.*KEY*.http-server-mechanism-factories

The referenced http server factories to aggregate.

swarm.elytron.aggregate-principal-decoders.*KEY*.principal-decoders

The referenced principal decoders to aggregate.

swarm.elytron.aggregate-principal-transformers.*KEY*.principal-transformers

The referenced principal transformers to aggregate.

swarm.elytron.aggregate-providers.*KEY*.providers

The referenced Provider[] resources to aggregate.

swarm.elytron.aggregate-realms.*KEY*.authentication-realm

Reference to the security realm to use for authentication steps (obtaining or validating credentials).

swarm.elytron.aggregate-realms.*KEY*.authorization-realm

Reference to the security realm to use for loading the identity for authorization steps (loading of the identity).

swarm.elytron.aggregate-role-mappers.*KEY*.role-mappers

The referenced role mappers to aggregate.

swarm.elytron.aggregate-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

swarm.elytron.aggregate-sasl-server-factories.*KEY*.sasl-server-factories

The referenced sasl server factories to aggregate.

swarm.elytron.aggregate-security-event-listeners.*KEY*.security-event-listeners

The referenced security event listener resources to aggregate.

swarm.elytron.authentication-configurations.*KEY*.anonymous

Enables anonymous authentication.

swarm.elytron.authentication-configurations.*KEY*.attribute-extends

A previously defined authentication configuration to extend.

swarm.elytron.authentication-configurations.*KEY*.authentication-name

The authentication name to use.

swarm.elytron.authentication-configurations.*KEY*.authorization-name

The authorization name to use.

swarm.elytron.authentication-configurations.*KEY*.credential-reference

The reference to credential stored in CredentialStore under defined alias or clear text password.

swarm.elytron.authentication-configurations.*KEY*.forwarding-mode

The type of security identity forwarding to use. A mode of 'authentication' forwarding forwards the principal and credential. A mode of 'authorization' forwards the authorization id, allowing for a different authentication identity.

swarm.elytron.authentication-configurations.*KEY*.host

The host to use.

swarm.elytron.authentication-configurations.*KEY*.kerberos-security-factory

Reference to a kerberos security factory used to obtain a GSS kerberos credential

swarm.elytron.authentication-configurations.*KEY*.mechanism-properties

Configuration properties for the SASL authentication mechanism.

swarm.elytron.authentication-configurations.*KEY*.port

The port to use.

swarm.elytron.authentication-configurations.*KEY*.protocol

The protocol to use.

swarm.elytron.authentication-configurations.*KEY*.realm

The realm to use.

swarm.elytron.authentication-configurations.*KEY*.sasl-mechanism-selector

The SASL mechanism selector string.

swarm.elytron.authentication-configurations.*KEY*.security-domain

Reference to a security domain to obtain a forwarded identity.

swarm.elytron.authentication-contexts.*KEY*.attribute-extends

A previously defined authentication context to extend.

swarm.elytron.authentication-contexts.*KEY*.match-rules

The match-rules for this authentication context.

swarm.elytron.caching-realms.*KEY*.maximum-age

The time in milliseconds that an item can stay in the cache.

swarm.elytron.caching-realms.*KEY*.maximum-entries

The maximum number of entries to keep in the cache.

swarm.elytron.caching-realms.*KEY*.realm

A reference to a cacheable security realm.

swarm.elytron.chained-principal-transformers.*KEY*.principal-transformers

The referenced principal transformers to chain.

swarm.elytron.client-ssl-contexts.*KEY*.active-session-count

The count of current active sessions.

swarm.elytron.client-ssl-contexts.*KEY*.cipher-suite-filter

The filter to apply to specify the enabled cipher suites.

swarm.elytron.client-ssl-contexts.*KEY*.key-manager

Reference to the key manager to use within the SSLContext.

swarm.elytron.client-ssl-contexts.*KEY*.protocols

The enabled protocols.

swarm.elytron.client-ssl-contexts.*KEY*.provider-name

The name of the provider to use. If not specified, all providers from providers will be passed to the SSLContext.

swarm.elytron.client-ssl-contexts.*KEY*.providers

The name of the providers to obtain the Provider[] to use to load the SSLContext.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.application-buffer-size

The application buffer size as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.cipher-suite

The selected cipher suite as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.creation-time

The creation time as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.last-accessed-time

The last accessed time as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.local-certificates

The local certificates from the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.local-principal

The local principal as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.packet-buffer-size

The packet buffer size as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-certificates

The peer certificates from the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-host

The peer host as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-port

The peer port as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-principal

The peer principal as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.protocol

The protocol as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.valid

The validity of the session as reported by the SSLSession.

swarm.elytron.client-ssl-contexts.*KEY*.trust-manager

Reference to the trust manager to use within the SSLContext.

swarm.elytron.concatenating-principal-decoders.*KEY*.joiner

The string to use to join the results of the referenced principal decoders.

swarm.elytron.concatenating-principal-decoders.*KEY*.principal-decoders

The referenced principal decoders to concatenate.

swarm.elytron.configurable-http-server-mechanism-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this factory instance.

swarm.elytron.configurable-http-server-mechanism-factories.*KEY*.filters

Filtering to be applied to enable / disable mechanisms based on the name.

swarm.elytron.configurable-http-server-mechanism-factories.*KEY*.http-server-mechanism-factory

The http server factory to be wrapped.

swarm.elytron.configurable-http-server-mechanism-factories.*KEY*.properties

Custom properties to be passed in to the http server factory calls.

swarm.elytron.configurable-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

swarm.elytron.configurable-sasl-server-factories.*KEY*.filters

List of filters to be evaluated sequentially combining the results using 'or'.

swarm.elytron.configurable-sasl-server-factories.*KEY*.properties

Custom properties to be passed in to the sasl server factory calls.

swarm.elytron.configurable-sasl-server-factories.*KEY*.protocol

The protocol that should be passed into factory when creating the mechanism.

swarm.elytron.configurable-sasl-server-factories.*KEY*.sasl-server-factory

The sasl server factory to be wrapped.

swarm.elytron.configurable-sasl-server-factories.*KEY*.server-name

The server name that should be passed into factory when creating the mechanism.

swarm.elytron.constant-permission-mappers.*KEY*.permissions

The permissions to assign.

swarm.elytron.constant-principal-decoders.*KEY*.constant

The constant value the principal decoder will always return.

swarm.elytron.constant-principal-transformers.*KEY*.constant

The constant value this PrincipalTransformer will always return.

swarm.elytron.constant-realm-mappers.*KEY*.realm-name

The name of the constant realm to return.

swarm.elytron.constant-role-mappers.*KEY*.roles

The constant roles to be returned by this role mapper.

swarm.elytron.credential-stores.*KEY*.create

Specifies whether credential store should create storage when it doesn’t exist.

swarm.elytron.credential-stores.*KEY*.credential-reference

Credential reference to be used to create protection parameter.

swarm.elytron.credential-stores.*KEY*.implementation-properties

Map of credentials store implementation specific properties.

swarm.elytron.credential-stores.*KEY*.location

File name of credential store storage.

swarm.elytron.credential-stores.*KEY*.modifiable

Specifies whether credential store is modifiable.

swarm.elytron.credential-stores.*KEY*.other-providers

The name of the providers defined within the subsystem to obtain the Providers to search for the one that can create the required JCA objects within credential store. This is valid only for key-store based CredentialStore. If this is not specified then the global list of Providers is used instead.

swarm.elytron.credential-stores.*KEY*.provider-name

The name of the provider to use to instantiate the CredentialStoreSpi. If the provider is not specified then the first provider found that can create an instance of the specified 'type' will be used.

swarm.elytron.credential-stores.*KEY*.providers

The name of the providers defined within the subsystem to obtain the Providers to search for the one that can create the required CredentialStore type. If this is not specified then the global list of Providers is used instead.

swarm.elytron.credential-stores.*KEY*.relative-to

A reference to a previously defined path that the file name is relative to.

swarm.elytron.credential-stores.*KEY*.state

The state of the underlying service that represents this credential store at runtime.

swarm.elytron.credential-stores.*KEY*.type

The credential store type, e.g. KeyStoreCredentialStore.

swarm.elytron.custom-credential-security-factories.*KEY*.class-name

The class name of the implementation of the custom security factory.

swarm.elytron.custom-credential-security-factories.*KEY*.configuration

The optional key/value configuration for the custom security factory.

swarm.elytron.custom-credential-security-factories.*KEY*.module

The module to use to load the custom security factory.

swarm.elytron.custom-modifiable-realms.*KEY*.class-name

The class name of the implementation of the custom realm.

swarm.elytron.custom-modifiable-realms.*KEY*.configuration

The optional key/value configuration for the custom realm.

swarm.elytron.custom-modifiable-realms.*KEY*.module

The module to use to load the custom realm.

swarm.elytron.custom-permission-mappers.*KEY*.class-name

Fully qualified class name of the permission mapper

swarm.elytron.custom-permission-mappers.*KEY*.configuration

The optional kay/value configuration for the permission mapper

swarm.elytron.custom-permission-mappers.*KEY*.module

Name of the module to use to load the permission mapper

swarm.elytron.custom-principal-decoders.*KEY*.class-name

Fully qualified class name of the principal decoder

swarm.elytron.custom-principal-decoders.*KEY*.configuration

The optional kay/value configuration for the principal decoder

swarm.elytron.custom-principal-decoders.*KEY*.module

Name of the module to use to load the principal decoder

swarm.elytron.custom-principal-transformers.*KEY*.class-name

The class name of the implementation of the custom principal transformer.

swarm.elytron.custom-principal-transformers.*KEY*.configuration

The optional key/value configuration for the custom principal transformer.

swarm.elytron.custom-principal-transformers.*KEY*.module

The module to use to load the custom principal transformer.

swarm.elytron.custom-realm-mappers.*KEY*.class-name

Fully qualified class name of the RealmMapper

swarm.elytron.custom-realm-mappers.*KEY*.configuration

The optional kay/value configuration for the RealmMapper

swarm.elytron.custom-realm-mappers.*KEY*.module

Name of the module to use to load the RealmMapper

swarm.elytron.custom-realms.*KEY*.class-name

The class name of the implementation of the custom realm.

swarm.elytron.custom-realms.*KEY*.configuration

The optional key/value configuration for the custom realm.

swarm.elytron.custom-realms.*KEY*.module

The module to use to load the custom realm.

swarm.elytron.custom-role-decoders.*KEY*.class-name

Fully qualified class name of the RoleDecoder

swarm.elytron.custom-role-decoders.*KEY*.configuration

The optional kay/value configuration for the RoleDecoder

swarm.elytron.custom-role-decoders.*KEY*.module

Name of the module to use to load the RoleDecoder

swarm.elytron.custom-role-mappers.*KEY*.class-name

Fully qualified class name of the RoleMapper

swarm.elytron.custom-role-mappers.*KEY*.configuration

The optional key/value configuration for the RoleMapper

swarm.elytron.custom-role-mappers.*KEY*.module

Name of the module to use to load the RoleMapper

swarm.elytron.default-authentication-context

The default authentication context to be associated with all deployments.

swarm.elytron.dir-contexts.*KEY*.authentication-context

The authentication context to obtain login credentials to connect to the LDAP server. Can be omitted if authentication-level is "none" (anonymous).

swarm.elytron.dir-contexts.*KEY*.authentication-level

The authentication level (security level/authentication mechanism) to use. Corresponds to SECURITY_AUTHENTICATION ("java.naming.security.authentication") environment property. Allowed values: "none", "simple", sasl_mech, where sasl_mech is a space-separated list of SASL mechanism names.

swarm.elytron.dir-contexts.*KEY*.connection-timeout

The timeout for connecting to the LDAP server in milliseconds.

swarm.elytron.dir-contexts.*KEY*.credential-reference

The credential reference to authenticate and connect to the LDAP server. Can be omitted if authentication-level is "none" (anonymous).

swarm.elytron.dir-contexts.*KEY*.enable-connection-pooling

Indicates if connection pooling is enabled.

swarm.elytron.dir-contexts.*KEY*.module

Name of module that will be used as class loading base.

swarm.elytron.dir-contexts.*KEY*.principal

The principal to authenticate and connect to the LDAP server. Can be omitted if authentication-level is "none" (anonymous).

swarm.elytron.dir-contexts.*KEY*.properties

The additional connection properties for the DirContext.

swarm.elytron.dir-contexts.*KEY*.read-timeout

The read timeout for an LDAP operation in milliseconds.

swarm.elytron.dir-contexts.*KEY*.referral-mode

If referrals should be followed.

swarm.elytron.dir-contexts.*KEY*.ssl-context

The name of ssl-context used to secure connection to the LDAP server.

swarm.elytron.dir-contexts.*KEY*.url

The connection url.

swarm.elytron.disallowed-providers

A list of providers that are not allowed, and will be removed from the providers list.

swarm.elytron.file-audit-logs.*KEY*.attribute-synchronized

Whether every event should be immediately synchronised to disk.

swarm.elytron.file-audit-logs.*KEY*.format

The format to use to record the audit event.

swarm.elytron.file-audit-logs.*KEY*.path

Path of the file to be written.

swarm.elytron.file-audit-logs.*KEY*.relative-to

The relative path to the audit log.

swarm.elytron.filesystem-realms.*KEY*.encoded

Whether the identity names should be stored encoded (Base32) in file names.

swarm.elytron.filesystem-realms.*KEY*.levels

The number of levels of directory hashing to apply.

swarm.elytron.filesystem-realms.*KEY*.path

The path to the file containing the realm.

swarm.elytron.filesystem-realms.*KEY*.relative-to

The pre-defined path the path is relative to.

swarm.elytron.filtering-key-stores.*KEY*.alias-filter

A filter to apply to the aliases returned from the KeyStore, can either be a comma separated list of aliases to return or one of the following formats ALL:-alias1:-alias2, NONE:+alias1:+alias2

swarm.elytron.filtering-key-stores.*KEY*.key-store

Name of filtered KeyStore.

swarm.elytron.filtering-key-stores.*KEY*.state

The state of the underlying service that represents this KeyStore at runtime, if it is anything other than UP runtime operations will not be available.

swarm.elytron.final-providers

Reference to the Providers that should be registered after all existing Providers.

swarm.elytron.http-authentication-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this configuration after all filtering has been applied.

swarm.elytron.http-authentication-factories.*KEY*.http-server-mechanism-factory

The HttpServerAuthenticationMechanismFactory to associate with this resource

swarm.elytron.http-authentication-factories.*KEY*.mechanism-configurations

Mechanism specific configuration

swarm.elytron.http-authentication-factories.*KEY*.security-domain

The SecurityDomain to associate with this resource

swarm.elytron.identity-realms.*KEY*.attribute-name

The name of the attribute associated with this identity.

swarm.elytron.identity-realms.*KEY*.attribute-values

The values associated with the identities attribute.

swarm.elytron.identity-realms.*KEY*.identity

The identity available from the security realm.

swarm.elytron.initial-providers

Reference to the Providers that should be registered ahead of all existing Providers.

swarm.elytron.jdbc-realms.*KEY*.principal-query

The authentication query used to authenticate users based on specific key types.

swarm.elytron.kerberos-security-factories.*KEY*.debug

Should the JAAS step of obtaining the credential have debug logging enabled.

swarm.elytron.kerberos-security-factories.*KEY*.mechanism-names

The mechanism names the credential should be usable with. Names will be converted to OIDs and used together with OIDs from mechanism-oids attribute.

swarm.elytron.kerberos-security-factories.*KEY*.mechanism-oids

The mechanism OIDs the credential should be usable with. Will be used together with OIDs derived from names from mechanism-names attribute.

swarm.elytron.kerberos-security-factories.*KEY*.minimum-remaining-lifetime

How much lifetime (in seconds) should a cached credential have remaining before it is recreated.

swarm.elytron.kerberos-security-factories.*KEY*.obtain-kerberos-ticket

Should the KerberosTicket also be obtained and associated with the credential. This is required to be true where credentials are delegated to the server.

swarm.elytron.kerberos-security-factories.*KEY*.options

The Krb5LoginModule additional options.

swarm.elytron.kerberos-security-factories.*KEY*.path

The path of the KeyTab to load to obtain the credential.

swarm.elytron.kerberos-security-factories.*KEY*.principal

The principal represented by the KeyTab

swarm.elytron.kerberos-security-factories.*KEY*.relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.elytron.kerberos-security-factories.*KEY*.request-lifetime

How much lifetime (in seconds) should be requested for newly created credentials.

swarm.elytron.kerberos-security-factories.*KEY*.required

Is the keytab file with adequate principal required to exist at the time the service starts?

swarm.elytron.kerberos-security-factories.*KEY*.server

If this for use server side or client side?

swarm.elytron.kerberos-security-factories.*KEY*.wrap-gss-credential

Should generated GSS credentials be wrapped to prevent improper disposal or not?

swarm.elytron.key-managers.*KEY*.algorithm

The name of the algorithm to use to create the underlying KeyManagerFactory.

swarm.elytron.key-managers.*KEY*.alias-filter

A filter to apply to the aliases returned from the KeyStore, can either be a comma separated list of aliases to return or one of the following formats ALL:-alias1:-alias2, NONE:+alias1:+alias2

swarm.elytron.key-managers.*KEY*.credential-reference

The credential reference to decrypt KeyStore item. (Not a password of the KeyStore.)

swarm.elytron.key-managers.*KEY*.key-store

Reference to the KeyStore to use to initialise the underlying KeyManagerFactory.

swarm.elytron.key-managers.*KEY*.provider-name

The name of the provider to use to create the underlying KeyManagerFactory.

swarm.elytron.key-managers.*KEY*.providers

Reference to obtain the Provider[] to use when creating the underlying KeyManagerFactory.

swarm.elytron.key-store-realms.*KEY*.key-store

Reference to the KeyStore that should be used to back this security realm.

swarm.elytron.key-stores.*KEY*.alias-filter

A filter to apply to the aliases returned from the KeyStore, can either be a comma separated list of aliases to return or one of the following formats ALL:-alias1:-alias2, NONE:+alias1:+alias2

swarm.elytron.key-stores.*KEY*.attribute-synchronized

The time this KeyStore was last loaded or saved. Note: Some providers may continue to apply updates after the KeyStore was loaded within the application server.

swarm.elytron.key-stores.*KEY*.credential-reference

The reference to credential stored in CredentialStore under defined alias or clear text password.

swarm.elytron.key-stores.*KEY*.loaded-provider

Information about the provider that was used for this KeyStore.

swarm.elytron.key-stores.*KEY*.modified

Indicates if the in-memory representation of the KeyStore has been changed since it was last loaded or stored. Note: For some providers updates may be immediate without further load or store calls.

swarm.elytron.key-stores.*KEY*.path

The path to the KeyStore file.

swarm.elytron.key-stores.*KEY*.provider-name

The name of the provider to use to load the KeyStore, disables searching for the first Provider that can create a KeyStore of the specified type.

swarm.elytron.key-stores.*KEY*.providers

A reference to the providers that should be used to obtain the list of Provider instances to search, if not specified the global list of providers will be used instead.

swarm.elytron.key-stores.*KEY*.relative-to

The base path this store is relative to.

swarm.elytron.key-stores.*KEY*.required

Is the file required to exist at the time the KeyStore service starts?

swarm.elytron.key-stores.*KEY*.size

The number of entries in the KeyStore.

swarm.elytron.key-stores.*KEY*.state

The state of the underlying service that represents this KeyStore at runtime, if it is anything other than UP runtime operations will not be available.

swarm.elytron.key-stores.*KEY*.type

The type of the KeyStore, used when creating the new KeyStore instance.

swarm.elytron.ldap-key-stores.*KEY*.alias-attribute

The name of LDAP attribute, where will be item alias stored.

swarm.elytron.ldap-key-stores.*KEY*.certificate-attribute

The name of LDAP attribute, where will be certificate stored.

swarm.elytron.ldap-key-stores.*KEY*.certificate-chain-attribute

The name of LDAP attribute, where will be certificate chain stored.

swarm.elytron.ldap-key-stores.*KEY*.certificate-chain-encoding

The encoding of the certificate chain.

swarm.elytron.ldap-key-stores.*KEY*.certificate-type

The type of the Certificate.

swarm.elytron.ldap-key-stores.*KEY*.dir-context

The name of DirContext, which will be used to communication with LDAP server.

swarm.elytron.ldap-key-stores.*KEY*.filter-alias

The LDAP filter for obtaining an item of the KeyStore by alias. If this is not specified then the default value will be (alias_attribute={0}). The string '{0}' will be replaced by the searched alias and the 'alias_attribute' value will be the value of the attribute 'alias-attribute'.

swarm.elytron.ldap-key-stores.*KEY*.filter-certificate

The LDAP filter for obtaining an item of the KeyStore by certificate. If this is not specified then the default value will be (certificate_attribute={0}). The string '{0}' will be replaced by searched encoded certificate and the 'certificate_attribute' will be the value of the attribute 'certificate-attribute'.

swarm.elytron.ldap-key-stores.*KEY*.filter-iterate

The LDAP filter for iterating over all items of the KeyStore. If this is not specified then the default value will be (alias_attribute=*). The 'alias_attribute' will be the value of the attribute 'alias-attribute'.

swarm.elytron.ldap-key-stores.*KEY*.key-attribute

The name of LDAP attribute, where will be key stored.

swarm.elytron.ldap-key-stores.*KEY*.key-type

The type of KeyStore, in which will be key serialized to LDAP attribute.

swarm.elytron.ldap-key-stores.*KEY*.new-item-template

Configuration for item creation. Define how will look LDAP entry of newly created keystore item.

swarm.elytron.ldap-key-stores.*KEY*.search-path

The path in LDAP, where will be KeyStore items searched.

swarm.elytron.ldap-key-stores.*KEY*.search-recursive

If the LDAP search should be recursive.

swarm.elytron.ldap-key-stores.*KEY*.search-time-limit

The time limit for obtaining keystore items from LDAP.

swarm.elytron.ldap-key-stores.*KEY*.size

The size of LDAP KeyStore in amount of items/aliases.

swarm.elytron.ldap-key-stores.*KEY*.state

The state of the underlying service that represents this KeyStore at runtime, if it is anything other than UP runtime operations will not be available.

swarm.elytron.ldap-realms.*KEY*.allow-blank-password

Does this realm support blank password direct verification? Blank password attempt will be rejected otherwise.

swarm.elytron.ldap-realms.*KEY*.dir-context

The configuration to connect to a LDAP server.

swarm.elytron.ldap-realms.*KEY*.direct-verification

Does this realm support verification of credentials by directly connecting to LDAP as the account being authenticated?

swarm.elytron.ldap-realms.*KEY*.identity-mapping

The configuration options that define how principals are mapped to their corresponding entries in the underlying LDAP server.

swarm.elytron.logical-permission-mappers.*KEY*.left

Reference to the permission mapper to use to the left of the operation.

swarm.elytron.logical-permission-mappers.*KEY*.logical-operation

The logical operation to use to combine the permission mappers.

swarm.elytron.logical-permission-mappers.*KEY*.right

Reference to the permission mapper to use to the right of the operation.

swarm.elytron.logical-role-mappers.*KEY*.left

Reference to a role mapper to be used on the left side of the operation.

swarm.elytron.logical-role-mappers.*KEY*.logical-operation

The logical operation to be performed on the role mapper mappings.

swarm.elytron.logical-role-mappers.*KEY*.right

Reference to a role mapper to be used on the right side of the operation.

swarm.elytron.mapped-regex-realm-mappers.*KEY*.delegate-realm-mapper

The RealmMapper to delegate to if the pattern does not match. If no delegate is specified then the default realm on the domain will be used instead. If the username does not match the pattern and a delegate realm-mapper is present, the result of delegate-realm-mapper is mapped via the realm-map.

swarm.elytron.mapped-regex-realm-mappers.*KEY*.pattern

The regular expression which must contain at least one capture group to extract the realm from the name. If the regular expression matches more than one capture group, the first capture group is used.

swarm.elytron.mapped-regex-realm-mappers.*KEY*.realm-map

Mapping of realm name extracted using the regular expression to a defined realm name. If the value for the mapping is not in the map or the realm whose name is the result of the mapping does not exist in the given security domain, the default realm is used.

swarm.elytron.mechanism-provider-filtering-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

swarm.elytron.mechanism-provider-filtering-sasl-server-factories.*KEY*.enabling

When set to 'true' no provider loaded mechanisms are enabled unless matched by one of the filters, setting to 'false' has the inverse effect.

swarm.elytron.mechanism-provider-filtering-sasl-server-factories.*KEY*.filters

The filters to apply when comparing the mechanisms from the providers, a filter matches when all of the specified values match the mechanism / provider pair.

swarm.elytron.mechanism-provider-filtering-sasl-server-factories.*KEY*.sasl-server-factory

Reference to a sasl server factory to be wrapped by this definition.

swarm.elytron.periodic-rotating-file-audit-logs.*KEY*.attribute-synchronized

Whether every event should be immediately synchronised to disk.

swarm.elytron.periodic-rotating-file-audit-logs.*KEY*.format

The format to use to record the audit event.

swarm.elytron.periodic-rotating-file-audit-logs.*KEY*.path

Path of the file to be written.

swarm.elytron.periodic-rotating-file-audit-logs.*KEY*.relative-to

The relative path to the audit log.

swarm.elytron.periodic-rotating-file-audit-logs.*KEY*.suffix

The suffix string in a format which can be understood by java.time.format.DateTimeFormatter. The period of the rotation is automatically calculated based on the suffix.

swarm.elytron.policies.*KEY*.custom-policy

A custom policy provider definition.

swarm.elytron.policies.*KEY*.jacc-policy

A policy provider definition that sets up JACC and related services.

swarm.elytron.properties-realms.*KEY*.attribute-synchronized

The time the properties files that back this realm were last loaded.

swarm.elytron.properties-realms.*KEY*.groups-attribute

The name of the attribute in the returned AuthorizationIdentity that should contain the group membership information for the identity.

swarm.elytron.properties-realms.*KEY*.groups-properties

The properties file containing the users and their groups.

swarm.elytron.properties-realms.*KEY*.users-properties

The properties file containing the users and their passwords.

swarm.elytron.provider-http-server-mechanism-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this factory instance.

swarm.elytron.provider-http-server-mechanism-factories.*KEY*.providers

The providers to use to locate the factories, if not specified the globally registered list of Providers will be used.

swarm.elytron.provider-loaders.*KEY*.argument

An argument to be passed into the constructor as the Provider is instantiated.

swarm.elytron.provider-loaders.*KEY*.class-names

The fully qualified class names of the providers to load, these are loaded after the service-loader discovered providers and duplicates will be skipped.

swarm.elytron.provider-loaders.*KEY*.configuration

The key/value configuration to be passed to the Provider to initialise it.

swarm.elytron.provider-loaders.*KEY*.loaded-providers

The list of providers loaded by this provider loader.

swarm.elytron.provider-loaders.*KEY*.module

The name of the module to load the provider from.

swarm.elytron.provider-loaders.*KEY*.path

The path of the file to use to initialise the providers.

swarm.elytron.provider-loaders.*KEY*.relative-to

The base path of the configuration file.

swarm.elytron.provider-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

swarm.elytron.provider-sasl-server-factories.*KEY*.providers

The providers to use to locate the factories, if not specified the globally registered list of Providers will be used.

swarm.elytron.regex-principal-transformers.*KEY*.pattern

The regular expression to use to locate the portion of the name to be replaced.

swarm.elytron.regex-principal-transformers.*KEY*.replace-all

Should all occurrences of the pattern matched be replaced or only the first occurrence.

swarm.elytron.regex-principal-transformers.*KEY*.replacement

The value to be used as the replacement.

swarm.elytron.regex-validating-principal-transformers.*KEY*.match

If set to true, the name must match the given pattern to make validation successful. If set to false, the name must not match the given pattern to make validation successful.

swarm.elytron.regex-validating-principal-transformers.*KEY*.pattern

The regular expression to use for the principal transformer.

swarm.elytron.sasl-authentication-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this configuration after all filtering has been applied.

swarm.elytron.sasl-authentication-factories.*KEY*.mechanism-configurations

Mechanism specific configuration

swarm.elytron.sasl-authentication-factories.*KEY*.sasl-server-factory

The SaslServerFactory to associate with this resource

swarm.elytron.sasl-authentication-factories.*KEY*.security-domain

The SecurityDomain to associate with this resource

swarm.elytron.security-domains.*KEY*.default-realm

The default realm contained by this security domain.

swarm.elytron.security-domains.*KEY*.outflow-anonymous

When outflowing to a security domain if outflow is not possible should the anonymous identity be used? Outflowing anonymous has the effect of clearing any identity already established for that domain.

swarm.elytron.security-domains.*KEY*.outflow-security-domains

The list of security domains that the security identity from this domain should automatically outflow to.

swarm.elytron.security-domains.*KEY*.permission-mapper

A reference to a PermissionMapper to be used by this domain.

swarm.elytron.security-domains.*KEY*.post-realm-principal-transformer

A reference to a principal transformer to be applied after the realm has operated on the supplied identity name.

swarm.elytron.security-domains.*KEY*.pre-realm-principal-transformer

A reference to a principal transformer to be applied before the realm is selected.

swarm.elytron.security-domains.*KEY*.principal-decoder

A reference to a PrincipalDecoder to be used by this domain.

swarm.elytron.security-domains.*KEY*.realm-mapper

Reference to the RealmMapper to be used by this domain.

swarm.elytron.security-domains.*KEY*.realms

The list of realms contained by this security domain.

swarm.elytron.security-domains.*KEY*.role-mapper

Reference to the RoleMapper to be used by this domain.

swarm.elytron.security-domains.*KEY*.security-event-listener

Reference to a listener for security events.

swarm.elytron.security-domains.*KEY*.trusted-security-domains

The list of security domains that are trusted by this security domain.

swarm.elytron.security-properties

Security properties to be set.

swarm.elytron.server-ssl-contexts.*KEY*.active-session-count

The count of current active sessions.

swarm.elytron.server-ssl-contexts.*KEY*.authentication-optional

Rejecting of the client certificate by the security domain will not prevent the connection. Allows a fall through to use other authentication mechanisms (like form login) when the client certificate is rejected by security domain. Has an effect only when the security domain is set.

swarm.elytron.server-ssl-contexts.*KEY*.cipher-suite-filter

The filter to apply to specify the enabled cipher suites.

swarm.elytron.server-ssl-contexts.*KEY*.final-principal-transformer

A final principal transformer to apply for this mechanism realm.

swarm.elytron.server-ssl-contexts.*KEY*.key-manager

Reference to the key manager to use within the SSLContext.

swarm.elytron.server-ssl-contexts.*KEY*.maximum-session-cache-size

The maximum number of SSL sessions in the cache. The default value -1 means use the JVM default value. Value zero means there is no limit.

swarm.elytron.server-ssl-contexts.*KEY*.need-client-auth

To require a client certificate on SSL handshake. Connection without trusted client certificate (see trust-manager) will be rejected.

swarm.elytron.server-ssl-contexts.*KEY*.post-realm-principal-transformer

A principal transformer to apply after the realm is selected.

swarm.elytron.server-ssl-contexts.*KEY*.pre-realm-principal-transformer

A principal transformer to apply before the realm is selected.

swarm.elytron.server-ssl-contexts.*KEY*.protocols

The enabled protocols.

swarm.elytron.server-ssl-contexts.*KEY*.provider-name

The name of the provider to use. If not specified, all providers from providers will be passed to the SSLContext.

swarm.elytron.server-ssl-contexts.*KEY*.providers

The name of the providers to obtain the Provider[] to use to load the SSLContext.

swarm.elytron.server-ssl-contexts.*KEY*.realm-mapper

The realm mapper to be used for SSL authentication.

swarm.elytron.server-ssl-contexts.*KEY*.security-domain

The security domain to use for authentication during SSL session establishment.

swarm.elytron.server-ssl-contexts.*KEY*.session-timeout

The timeout for SSL sessions, in seconds. The default value -1 means use the JVM default value. Value zero means there is no limit.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.application-buffer-size

The application buffer size as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.cipher-suite

The selected cipher suite as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.creation-time

The creation time as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.last-accessed-time

The last accessed time as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.local-certificates

The local certificates from the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.local-principal

The local principal as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.packet-buffer-size

The packet buffer size as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-certificates

The peer certificates from the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-host

The peer host as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-port

The peer port as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-principal

The peer principal as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.protocol

The protocol as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.valid

The validity of the session as reported by the SSLSession.

swarm.elytron.server-ssl-contexts.*KEY*.trust-manager

Reference to the trust manager to use within the SSLContext.

swarm.elytron.server-ssl-contexts.*KEY*.use-cipher-suites-order

To honor local cipher suites preference.

swarm.elytron.server-ssl-contexts.*KEY*.want-client-auth

To request (but not to require) a client certificate on SSL handshake. If a security domain is referenced and supports X509 evidence, this will be set to true automatically. Ignored when need-client-auth is set.

swarm.elytron.server-ssl-contexts.*KEY*.wrap

Should the SSLEngine, SSLSocket, and SSLServerSocket instances returned be wrapped to protect against further modification.

swarm.elytron.service-loader-http-server-mechanism-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this factory instance.

swarm.elytron.service-loader-http-server-mechanism-factories.*KEY*.module

The module to use to obtain the classloader to load the factories, if not specified the classloader to load the resource will be used instead.

swarm.elytron.service-loader-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

swarm.elytron.service-loader-sasl-server-factories.*KEY*.module

The module to use to obtain the classloader to load the factories, if not specified the classloader to load the resource will be used instead.

swarm.elytron.simple-permission-mappers.*KEY*.mapping-mode

The mapping mode that should be used in the event of multiple matches.

swarm.elytron.simple-permission-mappers.*KEY*.permission-mappings

The defined permission mappings.

swarm.elytron.simple-regex-realm-mappers.*KEY*.delegate-realm-mapper

The RealmMapper to delegate to if there is no match using the pattern.

swarm.elytron.simple-regex-realm-mappers.*KEY*.pattern

The regular expression which must contain at least one capture group to extract the realm from the name. If the regular expression matches more than one capture group, the first capture group is used.

swarm.elytron.simple-role-decoders.*KEY*.attribute

The name of the attribute from the identity to map directly to roles.

swarm.elytron.size-rotating-file-audit-logs.*KEY*.attribute-synchronized

Whether every event should be immediately synchronised to disk.

swarm.elytron.size-rotating-file-audit-logs.*KEY*.format

The format to use to record the audit event.

swarm.elytron.size-rotating-file-audit-logs.*KEY*.max-backup-index

The maximum number of files to backup when rotating.

swarm.elytron.size-rotating-file-audit-logs.*KEY*.path

Path of the file to be written.

swarm.elytron.size-rotating-file-audit-logs.*KEY*.relative-to

The relative path to the audit log.

swarm.elytron.size-rotating-file-audit-logs.*KEY*.rotate-on-boot

Whether the file should be rotated before the a new file is set.

swarm.elytron.size-rotating-file-audit-logs.*KEY*.rotate-size

The log file size the file should rotate at.

swarm.elytron.size-rotating-file-audit-logs.*KEY*.suffix

Format of date used as suffix of log file names in java.time.format.DateTimeFormatter. The suffix does not play a role in determining when the file should be rotated.

swarm.elytron.syslog-audit-logs.*KEY*.format

The format to use to record the audit event.

swarm.elytron.syslog-audit-logs.*KEY*.host-name

The host name to embed withing all events sent to the remote syslog server.

swarm.elytron.syslog-audit-logs.*KEY*.port

The listening port on the syslog server.

swarm.elytron.syslog-audit-logs.*KEY*.server-address

The server address of the syslog server the events should be sent to.

swarm.elytron.syslog-audit-logs.*KEY*.ssl-context

The SSLContext to use to connect to the syslog server when SSL_TCP transport is used.

swarm.elytron.syslog-audit-logs.*KEY*.transport

The transport to use to connect to the syslog server.

swarm.elytron.token-realms.*KEY*.jwt

A token validator to be used in conjunction with a token-based realm that handles security tokens based on the JWT/JWS standard.

swarm.elytron.token-realms.*KEY*.oauth2-introspection

A token validator to be used in conjunction with a token-based realm that handles OAuth2 Access Tokens and validates them using an endpoint compliant with OAuth2 Token Introspection specification(RFC-7662).

swarm.elytron.token-realms.*KEY*.principal-claim

The name of the claim that should be used to obtain the principal's name.

swarm.elytron.trust-managers.*KEY*.algorithm

The name of the algorithm to use to create the underlying TrustManagerFactory.

swarm.elytron.trust-managers.*KEY*.alias-filter

A filter to apply to the aliases returned from the KeyStore, can either be a comma separated list of aliases to return or one of the following formats ALL:-alias1:-alias2, NONE:+alias1:+alias2

swarm.elytron.trust-managers.*KEY*.certificate-revocation-list

Enables certificate revocation list checks to a trust manager.

swarm.elytron.trust-managers.*KEY*.key-store

Reference to the KeyStore to use to initialise the underlying TrustManagerFactory.

swarm.elytron.trust-managers.*KEY*.provider-name

The name of the provider to use to create the underlying TrustManagerFactory.

swarm.elytron.trust-managers.*KEY*.providers

Reference to obtain the Provider[] to use when creating the underlying TrustManagerFactory.

swarm.elytron.x500-attribute-principal-decoders.*KEY*.attribute-name

The name of the X.500 attribute to map (can be defined using OID instead)

swarm.elytron.x500-attribute-principal-decoders.*KEY*.convert

When set to 'true', if the Principal is not already an X500Principal conversion will be attempted

swarm.elytron.x500-attribute-principal-decoders.*KEY*.joiner

The joining string

swarm.elytron.x500-attribute-principal-decoders.*KEY*.maximum-segments

The maximum number of occurrences of the attribute to map

swarm.elytron.x500-attribute-principal-decoders.*KEY*.oid

The OID of the X.500 attribute to map (can be defined using attribute name instead)

swarm.elytron.x500-attribute-principal-decoders.*KEY*.required-attributes

The attributes names of the attributes that must be present in the principal

swarm.elytron.x500-attribute-principal-decoders.*KEY*.required-oids

The OIDs of the attributes that must be present in the principal

swarm.elytron.x500-attribute-principal-decoders.*KEY*.reverse

When set to 'true', the attribute values will be processed and returned in reverse order

swarm.elytron.x500-attribute-principal-decoders.*KEY*.start-segment

The 0-based starting occurrence of the attribute to map

D.10. HIBERNATE VALIDATOR

Provides support and integration for applications using Hibernate Validator.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>hibernate-validator</artifactId>
</dependency>
```

D.11. HYSTRIX

WARNING

This fraction is deprecated.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>hystrix</artifactId>
</dependency>
```

Configuration

swarm.hystrix.collapser.default.maxRequestsInBatch

The maximum number of requests allowed in a batch before this triggers a batch execution

swarm.hystrix.collapser.default.requestCache.enabled

Indicates whether request caching is enabled for HystrixCollapser.execute() and HystrixCollapser.queue() invocations

swarm.hystrix.collapser.default.timerDelayInMilliseconds

The number of milliseconds after the creation of the batch that its execution is triggered

swarm.hystrix.command.default.circuitBreaker.enabled

Determines whether a circuit breaker will be used to track health and to short-circuit requests if it trips

swarm.hystrix.command.default.circuitBreaker.errorThresholdPercentage

The error percentage at or above which the circuit should trip open and start short-circuiting requests to fallback logic

swarm.hystrix.command.default.circuitBreaker.forceClosed

If true, forces the circuit breaker into a closed state in which it will allow requests regardless of the error percentage

swarm.hystrix.command.default.circuitBreaker.forceOpen

If true, forces the circuit breaker into an open (tripped) state in which it will reject all requests

swarm.hystrix.command.default.circuitBreaker.requestVolumeThreshold

The minimum number of requests in a rolling window that will trip the circuit

swarm.hystrix.command.default.circuitBreaker.sleepWindowInMilliseconds

The amount of time, after tripping the circuit, to reject requests before allowing attempts again to determine if the circuit should again be closed

swarm.hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests

The maximum number of requests allowed to a HystrixCommand.run() method when you are using ExecutionIsolationStrategy.SEMAPHORE

swarm.hystrix.command.default.execution.isolation.strategy

Isolation strategy (THREAD or SEMAPHORE)

swarm.hystrix.command.default.execution.isolation.thread.interruptOnCancel

Indicates whether the HystrixCommand.run() execution should be interrupted when a cancellation occurs

swarm.hystrix.command.default.execution.isolation.thread.interruptOnTimeout

Indicates whether the HystrixCommand.run() execution should be interrupted when a timeout occurs

swarm.hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds

The time in milliseconds after which the caller will observe a timeout and walk away from the command execution

swarm.hystrix.command.default.execution.timeout.enabled

Indicates whether the HystrixCommand.run() execution should have a timeout

swarm.hystrix.command.default.fallback.enabled

Determines whether a call to HystrixCommand.getFallback() will be attempted when failure or rejection occurs

swarm.hystrix.command.default.fallback.isolation.semaphore.maxConcurrentRequests

The maximum number of requests allowed to a HystrixCommand.getFallback() method when you are using ExecutionIsolationStrategy.SEMAPHORE

swarm.hystrix.command.default.metrics.healthSnapshot.intervalInMilliseconds

The time to wait, in milliseconds, between allowing health snapshots to be taken that calculate success and error percentages and affect circuit breaker status

swarm.hystrix.command.default.metrics.rollingPercentile.bucketSize

The maximum number of execution times that are kept per bucket

swarm.hystrix.command.default.metrics.rollingPercentile.enabled

Indicates whether execution latencies should be tracked and calculated as percentiles

swarm.hystrix.command.default.metrics.rollingPercentile.numBuckets

The number of buckets the rollingPercentile window will be divided into

swarm.hystrix.command.default.metrics.rollingPercentile.timeInMilliseconds

The duration of the rolling window in which execution times are kept to allow for percentile calculations, in milliseconds

swarm.hystrix.command.default.metrics.rollingStats.numBuckets

The number of buckets the rolling statistical window is divided into

swarm.hystrix.command.default.metrics.rollingStats.timeInMilliseconds

The duration of the statistical rolling window, in milliseconds. This is how long Hystrix keeps metrics for the circuit breaker to use and for publishing

swarm.hystrix.command.default.requestCache.enabled

Indicates whether HystrixCommand.getCacheKey() should be used with HystrixRequestCache to provide de-duplication functionality via request-scoped caching

swarm.hystrix.command.default.requestLog.enabled

Indicates whether HystrixCommand execution and events should be logged to HystrixRequestLog

swarm.hystrix.stream.path

Context path for the stream

swarm.hystrix.threadpool.default.allowMaximumSizeToDivergeFromCoreSize

Allows the configuration for maximumSize to take effect

swarm.hystrix.threadpool.default.coreSize

The core thread-pool size

swarm.hystrix.threadpool.default.keepAliveTimeMinutes

The keep-alive time, in minutes

swarm.hystrix.threadpool.default.maxQueueSize

The maximum queue size of the BlockingQueue implementation

swarm.hystrix.threadpool.default.maximumSize

The maximum thread-pool size

swarm.hystrix.threadpool.default.metrics.rollingPercentile.numBuckets

The number of buckets the rolling statistical window is divided into

swarm.hystrix.threadpool.default.metrics.rollingStats.timeInMilliseconds

The duration of the statistical rolling window, in milliseconds

swarm.hystrix.threadpool.default.queueSizeRejectionThreshold

The queue size rejection threshold - an artificial maximum queue size at which rejections will occur even if maxQueueSize has not been reached

D.12. IO

Primarily an internal fraction supporting I/O activities for higher-level fractions.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>io</artifactId>
</dependency>
```

Configuration

swarm.io.buffer-pools.*KEY*.buffer-size

The size of each buffer slice in bytes, if not set optimal value is calculated based on available RAM resources in your system.

swarm.io.buffer-pools.*KEY*.buffers-per-slice

How many buffers per slice, if not set optimal value is calculated based on available RAM resources in your system.

swarm.io.buffer-pools.*KEY*.direct-buffers

Does the buffer pool use direct buffers, some platforms don't support direct buffers

swarm.io.workers.*KEY*.core-pool-size

Minimum number of threads to keep in the underlying java.util.concurrent.ThreadPoolExecutor even if they are idle. Threads over this limit will be terminated over time specified by task-keepalive attribute.

swarm.io.workers.*KEY*.io-thread-count

I/O thread count

swarm.io.workers.*KEY*.io-threads

Specify the number of I/O threads to create for the worker. If not specified, a default will be chosen, which is calculated by cpuCount * 2

swarm.io.workers.*KEY*.max-pool-size

The maximum number of threads to allow in the thread pool. Depending on implementation, when this limit is reached, tasks which cannot be queued may be rejected.

swarm.io.workers.*KEY*.outbound-bind-address.*KEY*.bind-address

The address to bind to when the destination address matches

swarm.io.workers.*KEY*.outbound-bind-address.*KEY*.bind-port

The port number to bind to when the destination address matches

swarm.io.workers.*KEY*.outbound-bind-address.*KEY*.match

The destination address range to match

swarm.io.workers.*KEY*.queue-size

An estimate of the number of tasks in the worker queue.

swarm.io.workers.*KEY*.servers.*KEY*.connection-count

Estimate of the current connection count

swarm.io.workers.*KEY*.servers.*KEY*.connection-limit-high-water-mark

If the connection count hits this number, no new connections will be accepted until the count drops below the low-water mark.

swarm.io.workers.*KEY*.servers.*KEY*.connection-limit-low-water-mark

If the connection count has previously hit the high water mark, once it drops back down below this count, connections will be accepted again.

swarm.io.workers.*KEY*.shutdown-requested

True is shutdown of the pool was requested

swarm.io.workers.*KEY*.stack-size

The stack size (in bytes) to attempt to use for worker threads.

swarm.io.workers.*KEY*.task-keepalive

Specify the number of milliseconds to keep non-core task threads alive.

swarm.io.workers.*KEY*.task-max-threads

Specify the maximum number of threads for the worker task thread pool.If not set, default value used which is calculated by formula cpuCount * 16,as long as MaxFileDescriptorCount jmx property allows that number, otherwise calculation takes max into account to adjust it accordingly.

D.13. JAEGER

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaeger</artifactId>
</dependency>
```

Configuration

swarm.jaeger.agent-host

The hostname for communicating with agent via UDP

swarm.jaeger.agent-port

The port for communicating with agent via UDP

swarm.jaeger.enable-b3-header-propagation

Whether to enable propagation of B3 headers in the configured Tracer. By default this is false.

swarm.jaeger.password

Password to send as part of "Basic" authentication to the endpoint

swarm.jaeger.remote-reporter-http-endpoint

Remote Reporter HTTP endpoint for Jaeger collector, such as http://jaeger-collector.istio-system:14268/api/traces

swarm.jaeger.reporter-flush-interval

The reporter's flush interval (ms)

swarm.jaeger.reporter-log-spans

Whether the reporter should also log the spans

swarm.jaeger.reporter-max-queue-size

The reporter's maximum queue size

swarm.jaeger.sampler-manager-host

The host name and port when using the remote controlled sampler

swarm.jaeger.sampler-parameter

The sampler parameter (number). Ex.: `1`

swarm.jaeger.sampler-type

The sampler type. Ex.: `const`

swarm.jaeger.service-name

The service name. Required (via this parameter, system property or env var). Ex.: `order-manager`

swarm.jaeger.user

Username to send as part of "Basic" authentication to the endpoint

D.14. JAX-RS

Provides support for building RESTful web services according to JSR-311.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaxrs</artifactId>
</dependency>
```

Configuration

swarm.deployment.*KEY*.jaxrs.application-path

Set the JAX-RS application path. If set, Thorntail will automatically generate a JAX-RS Application class and use this value as the @ApplicationPath

D.14.1. JAX-RS + CDI

An internal fraction providing integration between JAX-RS and CDI.

For more information, see the JAX-RS and CDI fraction documentation.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaxrs-cdi</artifactId>
</dependency>
```

D.14.2. JAX-RS + JAXB

Provides support within JAX-RS applications for the XML binding framework according to JSR-31 and JSR-222.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaxrs-jaxb</artifactId>
</dependency>
```

D.14.3. JAX-RS + JSON-P

Provides support within JAX-RS application for JSON processing according to JSR-374.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaxrs-jsonp</artifactId>
</dependency>
```

D.14.4. JAX-RS + Multipart

Provides support within JAX-RS application for MIME multipart form processing.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaxrs-multipart</artifactId>
</dependency>
```

D.14.5. JAX-RS + Validator

Provides integration and support between JAX-RS applications and Hibernate Validator.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaxrs-validator</artifactId>
</dependency>
```

D.15. JCA

Provides support for the Java Connector Architecture (JCA) according to JSR 322.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jca</artifactId>
</dependency>
```

Configuration

Configuration

swarm.jca.archive-validation.enabled

Specify whether archive validation is enabled

swarm.jca.archive-validation.fail-on-error

Should an archive validation error report fail the deployment

swarm.jca.archive-validation.fail-on-warn

Should an archive validation warning report fail the deployment

swarm.jca.bean-validation.enabled

Specify whether bean validation is enabled

swarm.jca.bootstrap-contexts.*KEY*.name

The name of the BootstrapContext

swarm.jca.bootstrap-contexts.*KEY*.workmanager

The WorkManager instance for the BootstrapContext

swarm.jca.cached-connection-manager.debug

Enable/disable debug information logging

swarm.jca.cached-connection-manager.error

Enable/disable error information logging

swarm.jca.cached-connection-manager.ignore-unknown-connections

Do not cache unknown connections

swarm.jca.cached-connection-manager.install

Enable/disable the cached connection manager valve and interceptor

swarm.jca.distributed-workmanagers.*KEY*.elytron-enabled

Enables Elytron security for this workmanager.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.allow-core-timeout

Whether core threads may time out.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.core-threads

The core thread pool size which is smaller than the maximum pool size. If undefined, the core thread pool size is the same as the maximum thread pool size.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.current-thread-count

The current number of threads in the pool.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.handoff-executor

An executor to delegate tasks to in the event that a task cannot be accepted. If not specified, tasks that cannot be accepted will be silently discarded.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.keepalive-time

Used to specify the amount of time that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.max-threads

The maximum thread pool size.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.name

The name of the thread pool.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.queue-length

The queue length.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.queue-size

The queue size.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.rejected-count

The number of tasks that have been passed to the handoff-executor (if one is specified) or discarded.

swarm.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.thread-factory

Specifies the name of a specific thread factory to use to create worker threads. If not defined an appropriate default thread factory will be used.

swarm.jca.distributed-workmanagers.*KEY*.name

The name of the DistributedWorkManager

swarm.jca.distributed-workmanagers.*KEY*.policy

The policy decides when to redistribute a Work instance

swarm.jca.distributed-workmanagers.*KEY*.policy-options

List of policy's options key/value pairs

swarm.jca.distributed-workmanagers.*KEY*.selector

The selector decides to which nodes in the network to redistribute the Work instance to

swarm.jca.distributed-workmanagers.*KEY*.selector-options

List of selector's options key/value pairs

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.allow-core-timeout

Whether core threads may time out.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.core-threads

The core thread pool size which is smaller than the maximum pool size. If undefined, the core thread pool size is the same as the maximum thread pool size.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.current-thread-count

The current number of threads in the pool.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.handoff-executor

An executor to delegate tasks to in the event that a task cannot be accepted. If not specified, tasks that cannot be accepted will be silently discarded.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.keepalive-time

Used to specify the amount of time that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.max-threads

The maximum thread pool size.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.name

The name of the thread pool.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.queue-length

The queue length.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.queue-size

The queue size.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.rejected-count

The number of tasks that have been passed to the handoff-executor (if one is specified) or discarded.

swarm.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.thread-factory

Specifies the name of a specific thread factory to use to create worker threads. If not defined an appropriate default thread factory will be used.

swarm.jca.tracer.enabled

Specify whether tracer is enabled

swarm.jca.workmanagers.*KEY*.elytron-enabled

Enables Elytron security for this workmanager.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.allow-core-timeout

Whether core threads may time out.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.core-threads

The core thread pool size which is smaller than the maximum pool size. If undefined, the core thread pool size is the same as the maximum thread pool size.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.current-thread-count

The current number of threads in the pool.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.handoff-executor

An executor to delegate tasks to in the event that a task cannot be accepted. If not specified, tasks that cannot be accepted will be silently discarded.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.keepalive-time

Used to specify the amount of time that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.max-threads

The maximum thread pool size.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.name

The name of the thread pool.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.queue-length

The queue length.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.queue-size

The queue size.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.rejected-count

The number of tasks that have been passed to the handoff-executor (if one is specified) or discarded.

swarm.jca.workmanagers.*KEY*.long-running-threads.*KEY*.thread-factory

Specifies the name of a specific thread factory to use to create worker threads. If not defined an appropriate default thread factory will be used.

swarm.jca.workmanagers.*KEY*.name

The name of the WorkManager

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.allow-core-timeout

Whether core threads may time out.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.core-threads

The core thread pool size which is smaller than the maximum pool size. If undefined, the core thread pool size is the same as the maximum thread pool size.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.current-thread-count

The current number of threads in the pool.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.handoff-executor

An executor to delegate tasks to in the event that a task cannot be accepted. If not specified, tasks that cannot be accepted will be silently discarded.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.keepalive-time

Used to specify the amount of time that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.max-threads

The maximum thread pool size.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.name

The name of the thread pool.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.queue-length

The queue length.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.queue-size

The queue size.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.rejected-count

The number of tasks that have been passed to the handoff-executor (if one is specified) or discarded.

swarm.jca.workmanagers.*KEY*.short-running-threads.*KEY*.thread-factory

Specifies the name of a specific thread factory to use to create worker threads. If not defined an appropriate default thread factory will be used.

D.16. JMX

Provides support for Java Management Extensions (JMX) according to JSR-3.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jmx</artifactId>
</dependency>
```

Configuration

swarm.jmx.audit-log-configuration.enabled

Whether audit logging is enabled.

swarm.jmx.audit-log-configuration.log-boot

Whether operations should be logged on server boot.

swarm.jmx.audit-log-configuration.log-read-only

Whether operations that do not modify the configuration or any runtime services should be logged.

swarm.jmx.expression-expose-model.domain-name

The domain name to use for the 'expression' model controller JMX facade in the MBeanServer.

swarm.jmx.jmx-remoting-connector.use-management-endpoint

If true the connector will use the management endpoint, otherwise it will use the remoting subsystem one

swarm.jmx.non-core-mbean-sensitivity

Whether or not core MBeans, i.e. mbeans not coming from the model controller, should be considered sensitive.

swarm.jmx.resolved-expose-model.domain-name

The domain name to use for the 'resolved' model controller JMX facade in the MBeanServer.

swarm.jmx.resolved-expose-model.proper-property-format

If false, PROPERTY type attributes are represented as a DMR string, this is the legacy behaviour. If true, PROPERTY type attributes are represented by a composite type where the key is a string, and the value has the same type as the property in the underlying model.

swarm.jmx.show-model

Alias for the existence of the 'resolved' model controller jmx facade. When writing, if set to 'true' it will add the 'resolved' model controller jmx facade resource with the default domain name.

D.17. JPA

Provides support for the Java Persistence API according to JSR-220.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jpa</artifactId>
</dependency>
```

Configuration

swarm.jpa.default-datasource

The name of the default global datasource.

swarm.jpa.default-extended-persistence-inheritance

Controls how JPA extended persistence context (XPC) inheritance is performed. 'DEEP' shares the extended persistence context at top bean level. 'SHALLOW' the extended persistece context is only shared with the parent bean (never with sibling beans).

D.18. JSF

Provides support for JavaServer Faces according to JSR-344.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jsf</artifactId>
</dependency>
```

Configuration

swarm.jsf.default-jsf-impl-slot

Default JSF implementation slot

D.19. JSON-P

Provides support for JSON Processing according to JSR-353.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jsonp</artifactId>
</dependency>
```

D.20. KEYCLOAK

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>keycloak</artifactId>
</dependency>
```

Configuration

swarm.keycloak.json.path

Path to Keycloak adapter configuration

swarm.keycloak.multitenancy.paths

Map of the relative request paths to Keycloak adapter configuration locations

swarm.keycloak.realms.*KEY*.allow-any-hostname

SSL Setting

swarm.keycloak.realms.*KEY*.always-refresh-token

Refresh token on every single web request

swarm.keycloak.realms.*KEY*.auth-server-url

Base URL of the Realm Auth Server

swarm.keycloak.realms.*KEY*.auth-server-url-for-backend-requests

URL to use to make background calls to auth server

swarm.keycloak.realms.*KEY*.autodetect-bearer-only

autodetect bearer-only requests

swarm.keycloak.realms.*KEY*.client-key-password

n/a

swarm.keycloak.realms.*KEY*.client-keystore

n/a

swarm.keycloak.realms.*KEY*.client-keystore-password

n/a

swarm.keycloak.realms.*KEY*.connection-pool-size

Connection pool size for the client used by the adapter

swarm.keycloak.realms.*KEY*.cors-allowed-headers

CORS allowed headers

swarm.keycloak.realms.*KEY*.cors-allowed-methods

CORS allowed methods

swarm.keycloak.realms.*KEY*.cors-exposed-headers

CORS exposed headers

swarm.keycloak.realms.*KEY*.cors-max-age

CORS max-age header

swarm.keycloak.realms.*KEY*.disable-trust-manager

Adapter will not use a trust manager when making adapter HTTPS requests

swarm.keycloak.realms.*KEY*.enable-cors

Enable Keycloak CORS support

swarm.keycloak.realms.*KEY*.expose-token

Enable secure URL that exposes access token

swarm.keycloak.realms.*KEY*.ignore-oauth-query-parameter

disable query parameter parsing for access_token

swarm.keycloak.realms.*KEY*.principal-attribute

token attribute to use to set Principal name

swarm.keycloak.realms.*KEY*.realm-public-key

Public key of the realm

swarm.keycloak.realms.*KEY*.register-node-at-startup

Cluster setting

swarm.keycloak.realms.*KEY*.register-node-period

how often to re-register node

swarm.keycloak.realms.*KEY*.ssl-required

Specify if SSL is required (valid values are all, external and none)

swarm.keycloak.realms.*KEY*.token-store

cookie or session storage for auth session data

swarm.keycloak.realms.*KEY*.truststore

Truststore used for adapter client HTTPS requests

swarm.keycloak.realms.*KEY*.truststore-password

Password of the Truststore

swarm.keycloak.secure-deployments.*KEY*.allow-any-hostname

SSL Setting

swarm.keycloak.secure-deployments.*KEY*.always-refresh-token

Refresh token on every single web request

swarm.keycloak.secure-deployments.*KEY*.auth-server-url

Base URL of the Realm Auth Server

swarm.keycloak.secure-deployments.*KEY*.auth-server-url-for-backend-requests

URL to use to make background calls to auth server

swarm.keycloak.secure-deployments.*KEY*.autodetect-bearer-only

autodetect bearer-only requests

swarm.keycloak.secure-deployments.*KEY*.bearer-only

Bearer Token Auth only

swarm.keycloak.secure-deployments.*KEY*.client-key-password

n/a

swarm.keycloak.secure-deployments.*KEY*.client-keystore

n/a

swarm.keycloak.secure-deployments.*KEY*.client-keystore-password

n/a

swarm.keycloak.secure-deployments.*KEY*.connection-pool-size

Connection pool size for the client used by the adapter

swarm.keycloak.secure-deployments.*KEY*.cors-allowed-headers

CORS allowed headers

swarm.keycloak.secure-deployments.*KEY*.cors-allowed-methods

CORS allowed methods

swarm.keycloak.secure-deployments.*KEY*.cors-exposed-headers

CORS exposed headers

swarm.keycloak.secure-deployments.*KEY*.cors-max-age

CORS max-age header

swarm.keycloak.secure-deployments.*KEY*.credentials.*KEY*.value

Credential value

swarm.keycloak.secure-deployments.*KEY*.disable-trust-manager

Adapter will not use a trust manager when making adapter HTTPS requests

swarm.keycloak.secure-deployments.*KEY*.enable-basic-auth

Enable Basic Authentication

swarm.keycloak.secure-deployments.*KEY*.enable-cors

Enable Keycloak CORS support

swarm.keycloak.secure-deployments.*KEY*.expose-token

Enable secure URL that exposes access token

swarm.keycloak.secure-deployments.*KEY*.ignore-oauth-query-parameter

disable query parameter parsing for access_token

swarm.keycloak.secure-deployments.*KEY*.min-time-between-jwks-requests

If adapter recognize token signed by unknown public key, it will try to download new public key from keycloak server. However it won’t try to download if already tried it in less than 'min-time-between-jwks-requests' seconds

swarm.keycloak.secure-deployments.*KEY*.principal-attribute

token attribute to use to set Principal name

swarm.keycloak.secure-deployments.*KEY*.public-client

Public client

swarm.keycloak.secure-deployments.*KEY*.realm

Keycloak realm

swarm.keycloak.secure-deployments.*KEY*.realm-public-key

Public key of the realm

swarm.keycloak.secure-deployments.*KEY*.redirect-rewrite-rules.*KEY*.value

redirect-rewrite-rule value

swarm.keycloak.secure-deployments.*KEY*.register-node-at-startup

Cluster setting

swarm.keycloak.secure-deployments.*KEY*.register-node-period

how often to re-register node

swarm.keycloak.secure-deployments.*KEY*.resource

Application name

swarm.keycloak.secure-deployments.*KEY*.ssl-required

Specify if SSL is required (valid values are all, external and none)

swarm.keycloak.secure-deployments.*KEY*.token-minimum-time-to-live

The adapter will refresh the token if the current token is expired OR will expire in 'token-minimum-time-to-live' seconds or less

swarm.keycloak.secure-deployments.*KEY*.token-store

cookie or session storage for auth session data

swarm.keycloak.secure-deployments.*KEY*.truststore

Truststore used for adapter client HTTPS requests

swarm.keycloak.secure-deployments.*KEY*.truststore-password

Password of the Truststore

swarm.keycloak.secure-deployments.*KEY*.turn-off-change-session-id-on-login

The session id is changed by default on a successful login. Change this to true if you want to turn this off

swarm.keycloak.secure-deployments.*KEY*.use-resource-role-mappings

Use resource level permissions from token

swarm.keycloak.secure-servers.*KEY*.allow-any-hostname

SSL Setting

swarm.keycloak.secure-servers.*KEY*.always-refresh-token

Refresh token on every single web request

swarm.keycloak.secure-servers.*KEY*.auth-server-url

Base URL of the Realm Auth Server

swarm.keycloak.secure-servers.*KEY*.auth-server-url-for-backend-requests

URL to use to make background calls to auth server

swarm.keycloak.secure-servers.*KEY*.autodetect-bearer-only

autodetect bearer-only requests

swarm.keycloak.secure-servers.*KEY*.bearer-only

Bearer Token Auth only

swarm.keycloak.secure-servers.*KEY*.client-key-password

n/a

swarm.keycloak.secure-servers.*KEY*.client-keystore

n/a

swarm.keycloak.secure-servers.*KEY*.client-keystore-password

n/a

swarm.keycloak.secure-servers.*KEY*.connection-pool-size

Connection pool size for the client used by the adapter

swarm.keycloak.secure-servers.*KEY*.cors-allowed-headers

CORS allowed headers

swarm.keycloak.secure-servers.*KEY*.cors-allowed-methods

CORS allowed methods

swarm.keycloak.secure-servers.*KEY*.cors-exposed-headers

CORS exposed headers

swarm.keycloak.secure-servers.*KEY*.cors-max-age

CORS max-age header

swarm.keycloak.secure-servers.*KEY*.credentials.*KEY*.value

Credential value

swarm.keycloak.secure-servers.*KEY*.disable-trust-manager

Adapter will not use a trust manager when making adapter HTTPS requests

swarm.keycloak.secure-servers.*KEY*.enable-basic-auth

Enable Basic Authentication

swarm.keycloak.secure-servers.*KEY*.enable-cors

Enable Keycloak CORS support

swarm.keycloak.secure-servers.*KEY*.expose-token

Enable secure URL that exposes access token

swarm.keycloak.secure-servers.*KEY*.ignore-oauth-query-parameter

disable query parameter parsing for access_token

swarm.keycloak.secure-servers.*KEY*.min-time-between-jwks-requests

If adapter recognize token signed by unknown public key, it will try to download new public key from keycloak server. However it won't try to download if already tried it in less than 'min-time-between-jwks-requests' seconds

swarm.keycloak.secure-servers.*KEY*.principal-attribute

token attribute to use to set Principal name

swarm.keycloak.secure-servers.*KEY*.public-client

Public client

swarm.keycloak.secure-servers.*KEY*.realm

Keycloak realm

swarm.keycloak.secure-servers.*KEY*.realm-public-key

Public key of the realm

swarm.keycloak.secure-servers.*KEY*.redirect-rewrite-rules.*KEY*.value

redirect-rewrite-rule value

swarm.keycloak.secure-servers.*KEY*.register-node-at-startup

Cluster setting

swarm.keycloak.secure-servers.*KEY*.register-node-period

how often to re-register node

swarm.keycloak.secure-servers.*KEY*.resource

Application name

swarm.keycloak.secure-servers.*KEY*.ssl-required

Specify if SSL is required (valid values are all, external and none)

swarm.keycloak.secure-servers.*KEY*.token-minimum-time-to-live

The adapter will refresh the token if the current token is expired OR will expire in 'token-minimum-time-to-live' seconds or less

swarm.keycloak.secure-servers.*KEY*.token-store

cookie or session storage for auth session data

swarm.keycloak.secure-servers.*KEY*.truststore

Truststore used for adapter client HTTPS requests

swarm.keycloak.secure-servers.*KEY*.truststore-password

Password of the Truststore

swarm.keycloak.secure-servers.*KEY*.turn-off-change-session-id-on-login

The session id is changed by default on a successful login. Change this to true if you want to turn this off

swarm.keycloak.secure-servers.*KEY*.use-resource-role-mappings

Use resource level permissions from token

D.21. LOGGING

Provides facilities to configure logging categories, levels and handlers.

When specifying log-levels through properties, since they include dots, they should be placed between square brackets, such as `swarm.logging.loggers.[com.mycorp.logger].level`.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>logging</artifactId>
</dependency>
```

Configuration

swarm.logging.add-logging-api-dependencies

Indicates whether or not logging API dependencies should be added to deployments during the deployment process. A value of true will add the dependencies to the deployment. A value of false will skip the deployment from being processed for logging API dependencies.

swarm.logging.async-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.async-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.async-handlers.*KEY*.level

The log level specifying which message levels will be logged by this handler. Message levels lower than this value will be discarded.

swarm.logging.async-handlers.*KEY*.name

The name of the handler.

swarm.logging.async-handlers.*KEY*.overflow-action

Specify what action to take when the overflowing. The valid options are 'block' and 'discard'

swarm.logging.async-handlers.*KEY*.queue-length

The queue length to use before flushing writing

swarm.logging.async-handlers.*KEY*.subhandlers

The Handlers associated with this async handler.

swarm.logging.console-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.console-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.console-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.console-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.console-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.console-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.console-handlers.*KEY*.name

The name of the handler.

swarm.logging.console-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.console-handlers.*KEY*.target

Defines the target of the console handler. The value can be System.out, System.err or console.

swarm.logging.custom-formatters.*KEY*.attribute-class

The logging handler class to be used.

swarm.logging.custom-formatters.*KEY*.module

The module that the logging handler depends on.

swarm.logging.custom-formatters.*KEY*.properties

Defines the properties used for the logging handler. All properties must be accessible via a setter method.

swarm.logging.custom-handlers.*KEY*.attribute-class

The logging handler class to be used.

swarm.logging.custom-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.custom-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.custom-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.custom-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.custom-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.custom-handlers.*KEY*.module

The module that the logging handler depends on.

swarm.logging.custom-handlers.*KEY*.name

The name of the handler.

swarm.logging.custom-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.custom-handlers.*KEY*.properties

Defines the properties used for the logging handler. All properties must be accessible via a setter method.

swarm.logging.file-handlers.*KEY*.append

Specify whether to append to the target file.

swarm.logging.file-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.file-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.file-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

swarm.logging.file-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.file-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.file-handlers.*KEY*.name

The name of the handler.

swarm.logging.file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.log-files.*KEY*.file-size

The size of the log file in bytes.

swarm.logging.log-files.*KEY*.last-modified-time

The date, in milliseconds, the file was last modified.

swarm.logging.log-files.*KEY*.last-modified-timestamp

The date, in ISO 8601 format, the file was last modified.

swarm.logging.log-files.*KEY*.stream

Provides the server log as a response attachment. The response result value is the unique id of the attachment.

swarm.logging.loggers.*KEY*.category

Specifies the category for the logger.

swarm.logging.loggers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.loggers.*KEY*.handlers

The handlers associated with the logger.

swarm.logging.loggers.*KEY*.level

The log level specifying which message levels will be logged by the logger. Message levels lower than this value will be discarded.

swarm.logging.loggers.*KEY*.use-parent-handlers

Specifies whether or not this logger should send its output to its parent Logger.

swarm.logging.logging-profiles.*KEY*.async-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.logging-profiles.*KEY*.async-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.async-handlers.*KEY*.level

The log level specifying which message levels will be logged by this handler. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.async-handlers.*KEY*.name

The name of the handler.

swarm.logging.logging-profiles.*KEY*.async-handlers.*KEY*.overflow-action

Specify what action to take when the overflowing. The valid options are 'block' and 'discard'

swarm.logging.logging-profiles.*KEY*.async-handlers.*KEY*.queue-length

The queue length to use before flushing writing

swarm.logging.logging-profiles.*KEY*.async-handlers.*KEY*.subhandlers

The Handlers associated with this async handler.

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.name

The name of the handler.

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.logging-profiles.*KEY*.console-handlers.*KEY*.target

Defines the target of the console handler. The value can be System.out, System.err or console.

swarm.logging.logging-profiles.*KEY*.custom-formatters.*KEY*.attribute-class

The logging handler class to be used.

swarm.logging.logging-profiles.*KEY*.custom-formatters.*KEY*.module

The module that the logging handler depends on.

swarm.logging.logging-profiles.*KEY*.custom-formatters.*KEY*.properties

Defines the properties used for the logging handler. All properties must be accessible via a setter method.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.attribute-class

The logging handler class to be used.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.module

The module that the logging handler depends on.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.name

The name of the handler.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.properties

Defines the properties used for the logging handler. All properties must be accessible via a setter method.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.append

Specify whether to append to the target file.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.name

The name of the handler.

swarm.logging.logging-profiles.*KEY*.file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.logging-profiles.*KEY*.log-files.*KEY*.file-size

The size of the log file in bytes.

swarm.logging.logging-profiles.*KEY*.log-files.*KEY*.last-modified-time

The date, in milliseconds, the file was last modified.

swarm.logging.logging-profiles.*KEY*.log-files.*KEY*.last-modified-timestamp

The date, in ISO 8601 format, the file was last modified.

swarm.logging.logging-profiles.*KEY*.log-files.*KEY*.stream

Provides the server log as a response attachment. The response result value is the unique id of the attachment.

swarm.logging.logging-profiles.*KEY*.loggers.*KEY*.category

Specifies the category for the logger.

swarm.logging.logging-profiles.*KEY*.loggers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.loggers.*KEY*.handlers

The handlers associated with the logger.

swarm.logging.logging-profiles.*KEY*.loggers.*KEY*.level

The log level specifying which message levels will be logged by the logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.loggers.*KEY*.use-parent-handlers

Specifies whether or not this logger should send its output to its parent Logger.

swarm.logging.logging-profiles.*KEY*.pattern-formatters.*KEY*.color-map

The color-map attribute allows for a comma delimited list of colors to be used for different levels with a pattern formatter. The format for the color mapping pattern is level-name:color-name.Valid Levels; severe, fatal, error, warn, warning, info, debug, trace, config, fine, finer, finest Valid Colors; black, green, red, yellow, blue, magenta, cyan, white, brightblack, brightred, brightgreen, brightblue, brightyellow, brightmagenta, brightcyan, brightwhite

swarm.logging.logging-profiles.*KEY*.pattern-formatters.*KEY*.pattern

Defines a pattern for the formatter.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.name

The name of the handler.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.suffix

Set the suffix string. The string is in a format which can be understood by java.text.SimpleDateFormat. The period of the rotation is automatically calculated based on the suffix.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.name

The name of the handler.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.rotate-on-boot

Indicates the file should be rotated each time the file attribute is changed. This always happens when at initialization time.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

swarm.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.suffix

Set the suffix string. The string is in a format which can be understood by java.text.SimpleDateFormat. The period of the rotation is automatically calculated based on the suffix.

swarm.logging.logging-profiles.*KEY*.root-logger.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.root-logger.handlers

The handlers associated with the root logger.

swarm.logging.logging-profiles.*KEY*.root-logger.level

The log level specifying which message levels will be logged by the root logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.name

The name of the handler.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.rotate-on-boot

Indicates the file should be rotated each time the file attribute is changed. This always happens when at initialization time.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

swarm.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.suffix

Set the suffix string. The string is in a format which can be understood by java.text.SimpleDateFormat. The suffix does not determine when the file should be rotated.

swarm.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.app-name

The app name used when formatting the message in RFC5424 format. By default the app name is "java".

swarm.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.facility

Facility as defined by RFC-5424 (http://tools.ietf.org/html/rfc5424)and RFC-3164 (http://tools.ietf.org/html/rfc3164).

swarm.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.hostname

The name of the host the messages are being sent from. For example the name of the host the application server is running on.

swarm.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.port

The port the syslog server is listening on.

swarm.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.server-address

The address of the syslog server.

swarm.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.syslog-format

Formats the log message according to the RFC specification.

swarm.logging.pattern-formatters.*KEY*.color-map

The color-map attribute allows for a comma delimited list of colors to be used for different levels with a pattern formatter. The format for the color mapping pattern is level-name:color-name.Valid Levels; severe, fatal, error, warn, warning, info, debug, trace, config, fine, finer, finest Valid Colors; black, green, red, yellow, blue, magenta, cyan, white, brightblack, brightred, brightgreen, brightblue, brightyellow, brightmagenta, brightcyan, brightwhite

swarm.logging.pattern-formatters.*KEY*.pattern

Defines a pattern for the formatter.

swarm.logging.periodic-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

swarm.logging.periodic-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.periodic-rotating-file-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.periodic-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.periodic-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

swarm.logging.periodic-rotating-file-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.periodic-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.periodic-rotating-file-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.periodic-rotating-file-handlers.*KEY*.name

The name of the handler.

swarm.logging.periodic-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.periodic-rotating-file-handlers.*KEY*.suffix

Set the suffix string. The string is in a format which can be understood by java.text.SimpleDateFormat. The period of the rotation is automatically calculated based on the suffix.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.name

The name of the handler.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.rotate-on-boot

Indicates the file should be rotated each time the file attribute is changed. This always happens when at initialization time.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

swarm.logging.periodic-size-rotating-file-handlers.*KEY*.suffix

Set the suffix string. The string is in a format which can be understood by java.text.SimpleDateFormat. The period of the rotation is automatically calculated based on the suffix.

swarm.logging.root-logger.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.root-logger.handlers

The handlers associated with the root logger.

swarm.logging.root-logger.level

The log level specifying which message levels will be logged by the root logger. Message levels lower than this value will be discarded.

swarm.logging.size-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

swarm.logging.size-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

swarm.logging.size-rotating-file-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.size-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

swarm.logging.size-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

swarm.logging.size-rotating-file-handlers.*KEY*.filter-spec

A filter expression value to define a filter. Example for a filter that does not match a pattern: not(match("JBAS.*"))

swarm.logging.size-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

swarm.logging.size-rotating-file-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

swarm.logging.size-rotating-file-handlers.*KEY*.name

The name of the handler.

swarm.logging.size-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

swarm.logging.size-rotating-file-handlers.*KEY*.rotate-on-boot

Indicates the file should be rotated each time the file attribute is changed. This always happens when at initialization time.

swarm.logging.size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

swarm.logging.size-rotating-file-handlers.*KEY*.suffix

Set the suffix string. The string is in a format which can be understood by java.text.SimpleDateFormat. The suffix does not determine when the file should be rotated.

swarm.logging.syslog-handlers.*KEY*.app-name

The app name used when formatting the message in RFC5424 format. By default the app name is "java".

swarm.logging.syslog-handlers.*KEY*.enabled

If set to true the handler is enabled and functioning as normal, if set to false the handler is ignored when processing log messages.

swarm.logging.syslog-handlers.*KEY*.facility

Facility as defined by RFC-5424 (http://tools.ietf.org/html/rfc5424)and RFC-3164 (http://tools.ietf.org/html/rfc3164).

swarm.logging.syslog-handlers.*KEY*.hostname

The name of the host the messages are being sent from. For example the name of the host the application server is running on.

swarm.logging.syslog-handlers.*KEY*.level

The log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded.

swarm.logging.syslog-handlers.*KEY*.port

The port the syslog server is listening on.

swarm.logging.syslog-handlers.*KEY*.server-address

The address of the syslog server.

swarm.logging.syslog-handlers.*KEY*.syslog-format

Formats the log message according to the RFC specification.

swarm.logging.use-deployment-logging-config

Indicates whether or not deployments should use a logging configuration file found in the deployment to configure the log manager. If set to true and a logging configuration file was found in the deployments META-INF or WEB-INF/classes directory, then a log manager will be configured with those settings. If set false the servers logging configuration will be used regardless of any logging configuration files supplied in the deployment.

D.22. MANAGEMENT

Provides the JBoss EAP management API.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>management</artifactId>
</dependency>
```

Configuration

swarm.management.audit-access.audit-log-logger.enabled

Whether audit logging is enabled.

swarm.management.audit-access.audit-log-logger.log-boot

Whether operations should be logged on server boot.

swarm.management.audit-access.audit-log-logger.log-read-only

Whether operations that do not modify the configuration or any runtime services should be logged.

swarm.management.audit-access.file-handlers.*KEY*.disabled-due-to-failure

Whether this handler has been disabled due to logging failures.

swarm.management.audit-access.file-handlers.*KEY*.failure-count

The number of logging failures since the handler was initialized.

swarm.management.audit-access.file-handlers.*KEY*.formatter

The formatter used to format the log messages.

swarm.management.audit-access.file-handlers.*KEY*.max-failure-count

The maximum number of logging failures before disabling this handler.

swarm.management.audit-access.file-handlers.*KEY*.path

The path of the audit log file.

swarm.management.audit-access.file-handlers.*KEY*.relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.management.audit-access.file-handlers.*KEY*.rotate-at-startup

Whether the old log file should be rotated at server startup.

swarm.management.audit-access.in-memory-handlers.*KEY*.max-history

The maximum number of operation stored in history for this handler.

swarm.management.audit-access.json-formatters.*KEY*.compact

If true will format the JSON on one line. There may still be values containing new lines, so if having the whole record on one line is important, set escape-new-line or escape-control-characters to true.

swarm.management.audit-access.json-formatters.*KEY*.date-format

The date format to use as understood by java.text.SimpleDateFormat. Will be ignored if include-date="false".

swarm.management.audit-access.json-formatters.*KEY*.date-separator

The separator between the date and the rest of the formatted log message. Will be ignored if include-date="false".

swarm.management.audit-access.json-formatters.*KEY*.escape-control-characters

If true will escape all control characters (ascii entries with a decimal value < 32) with the ascii code in octal, e.g.' becomes '#012'. If this is true, it will override escape-new-line="false".

swarm.management.audit-access.json-formatters.*KEY*.escape-new-line

If true will escape all new lines with the ascii code in octal, e.g. "#012".

swarm.management.audit-access.json-formatters.*KEY*.include-date

Whether or not to include the date in the formatted log record.

swarm.management.audit-access.periodic-rotating-file-handlers.*KEY*.disabled-due-to-failure

Whether this handler has been disabled due to logging failures.

swarm.management.audit-access.periodic-rotating-file-handlers.*KEY*.failure-count

The number of logging failures since the handler was initialized.

swarm.management.audit-access.periodic-rotating-file-handlers.*KEY*.formatter

The formatter used to format the log messages.

swarm.management.audit-access.periodic-rotating-file-handlers.*KEY*.max-failure-count

The maximum number of logging failures before disabling this handler.

swarm.management.audit-access.periodic-rotating-file-handlers.*KEY*.path

The path of the audit log file.

swarm.management.audit-access.periodic-rotating-file-handlers.*KEY*.relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.management.audit-access.periodic-rotating-file-handlers.*KEY*.suffix

The suffix string in a format which can be understood by java.text.SimpleDateFormat. The period of the rotation is automatically calculated based on the suffix.

swarm.management.audit-access.size-rotating-file-handlers.*KEY*.disabled-due-to-failure

Whether this handler has been disabled due to logging failures.

swarm.management.audit-access.size-rotating-file-handlers.*KEY*.failure-count

The number of logging failures since the handler was initialized.

swarm.management.audit-access.size-rotating-file-handlers.*KEY*.formatter

The formatter used to format the log messages.

swarm.management.audit-access.size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

swarm.management.audit-access.size-rotating-file-handlers.*KEY*.max-failure-count

The maximum number of logging failures before disabling this handler.

swarm.management.audit-access.size-rotating-file-handlers.*KEY*.path

The path of the audit log file.

swarm.management.audit-access.size-rotating-file-handlers.*KEY*.relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.management.audit-access.size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

swarm.management.audit-access.syslog-handlers.*KEY*.app-name

The application name to add to the syslog records as defined in section 6.2.5 of RFC-5424. If not specified it will default to the name of the product.

swarm.management.audit-access.syslog-handlers.*KEY*.disabled-due-to-failure

Whether this handler has been disabled due to logging failures.

swarm.management.audit-access.syslog-handlers.*KEY*.facility

The facility to use for syslog logging as defined in section 6.2.1 of RFC-5424, and section 4.1.1 of RFC-3164.

swarm.management.audit-access.syslog-handlers.*KEY*.failure-count

The number of logging failures since the handler was initialized.

swarm.management.audit-access.syslog-handlers.*KEY*.formatter

The formatter used to format the log messages.

swarm.management.audit-access.syslog-handlers.*KEY*.max-failure-count

The maximum number of logging failures before disabling this handler.

swarm.management.audit-access.syslog-handlers.*KEY*.max-length

The maximum length in bytes a log message, including the header, is allowed to be. If undefined, it will default to 1024 bytes if the syslog-format is RFC3164, or 2048 bytes if the syslog-format is RFC5424.

swarm.management.audit-access.syslog-handlers.*KEY*.syslog-format

Whether to set the syslog format to the one specified in RFC-5424 or RFC-3164.

swarm.management.audit-access.syslog-handlers.*KEY*.tcp-protocol.host

The host of the syslog server for the tcp requests.

swarm.management.audit-access.syslog-handlers.*KEY*.tcp-protocol.message-transfer

The message transfer setting as described in section 3.4 of RFC-6587. This can either be OCTET_COUNTING as described in section 3.4.1 of RFC-6587, or NON_TRANSPARENT_FRAMING as described in section 3.4.1 of RFC-6587. See your syslog provider's documentation for what is supported.

swarm.management.audit-access.syslog-handlers.*KEY*.tcp-protocol.port

The port of the syslog server for the tcp requests.

swarm.management.audit-access.syslog-handlers.*KEY*.tcp-protocol.reconnect-timeout

If a connection drop is detected, the number of seconds to wait before reconnecting. A negative number means don't reconnect automatically.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.key-password

The password for the keystore key.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.key-password-credential-reference

The reference to credential for the keystore key stored in CredentialStore under defined alias or clear text password.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.keystore-password

The password for the keystore.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.keystore-password-credential-reference

The reference to credential for the keystore password stored in CredentialStore under defined alias or clear text password.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.keystore-path

The path of the keystore.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.keystore-relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'keystore-relative-to' is provided, the value of the 'keystore-path' attribute is treated as relative to the path specified by this attribute.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.host

The host of the syslog server for the tls over tcp requests.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.message-transfer

The message transfer setting as described in section 3.4 of RFC-6587. This can either be OCTET_COUNTING as described in section 3.4.1 of RFC-6587, or NON_TRANSPARENT_FRAMING as described in section 3.4.1 of RFC-6587. See your syslog provider’s documentation for what is supported.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.port

The port of the syslog server for the tls over tcp requests.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.reconnect-timeout

If a connection drop is detected, the number of seconds to wait before reconnecting. A negative number means don’t reconnect automatically.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.truststore-authentication.keystore-password

The password for the truststore.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.truststore-authentication.keystore-password-credential-reference

The reference to credential for the truststore password stored in CredentialStore under defined alias or clear text password.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.truststore-authentication.keystore-path

The path of the truststore.

swarm.management.audit-access.syslog-handlers.*KEY*.tls-protocol.truststore-authentication.keystore-relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'keystore-relative-to' is provided, the value of the 'keystore-path' attribute is treated as relative to the path specified by this attribute.

swarm.management.audit-access.syslog-handlers.*KEY*.truncate

Whether or not a message, including the header, should truncate the message if the length in bytes is greater than the maximum length. If set to false messages will be split and sent with the same header values.

swarm.management.audit-access.syslog-handlers.*KEY*.udp-protocol.host

The host of the syslog server for the udp requests.

swarm.management.audit-access.syslog-handlers.*KEY*.udp-protocol.port

The port of the syslog server for the udp requests.

swarm.management.authorization-access.all-role-names

The official names of all roles supported by the current management access control provider. This includes any standard roles as well as any user-defined roles.

swarm.management.authorization-access.application-classification-constraint.types.*KEY*.classifications.*KEY*.applies-tos.*KEY*.address

Address pattern describing a resource or resources to which the constraint applies.

swarm.management.authorization-access.application-classification-constraint.types.*KEY*.classifications.*KEY*.applies-tos.*KEY*.attributes

List of the names of attributes to which the constraint specifically applies.

swarm.management.authorization-access.application-classification-constraint.types.*KEY*.classifications.*KEY*.applies-tos.*KEY*.entire-resource

True if the constraint applies to the resource as a whole; false if it only applies to one or more attributes or operations.

swarm.management.authorization-access.application-classification-constraint.types.*KEY*.classifications.*KEY*.applies-tos.*KEY*.operations

List of the names of operations to which the constraint specifically applies.

swarm.management.authorization-access.application-classification-constraint.types.*KEY*.classifications.*KEY*.configured-application

Set to override the default as to whether the constraint is considered an application resource.

swarm.management.authorization-access.application-classification-constraint.types.*KEY*.classifications.*KEY*.default-application

Whether targets having this application type constraint are considered application resources.

swarm.management.authorization-access.permission-combination-policy

The policy for combining access control permissions when the authorization policy grants the user more than one type of permission for a given action. In the standard role based authorization policy, this would occur when a user maps to multiple roles. The 'permissive' policy means if any of the permissions allow the action, the action is allowed. The 'rejecting' policy means the existence of multiple permissions should result in an error.

swarm.management.authorization-access.provider

The provider to use for management access control decisions.

swarm.management.authorization-access.role-mappings.*KEY*.excludes.*KEY*.name

The name of the user or group being mapped.

swarm.management.authorization-access.role-mappings.*KEY*.excludes.*KEY*.realm

An optional attribute to map based on the realm used for authentication.

swarm.management.authorization-access.role-mappings.*KEY*.excludes.*KEY*.type

The type of the Principal being mapped, either 'group' or 'user'.

swarm.management.authorization-access.role-mappings.*KEY*.include-all

Configure if all authenticated users should be automatically assigned this role.

swarm.management.authorization-access.role-mappings.*KEY*.includes.*KEY*.name

The name of the user or group being mapped.

swarm.management.authorization-access.role-mappings.*KEY*.includes.*KEY*.realm

An optional attribute to map based on the realm used for authentication.

swarm.management.authorization-access.role-mappings.*KEY*.includes.*KEY*.type

The type of the Principal being mapped, either 'group' or 'user'.

swarm.management.authorization-access.sensitivity-classification-constraint.types.*KEY*.classifications.*KEY*.applies-tos.*KEY*.address

Address pattern describing a resource or resources to which the constraint applies.

swarm.management.authorization-access.sensitivity-classification-constraint.types.*KEY*.classifications.*KEY*.applies-tos.*KEY*.attributes

List of the names of attributes to which the constraint specifically applies.

swarm.management.authorization-access.sensitivity-classification-constraint.types.*KEY*.classifications.*KEY*.applies-tos.*KEY*.entire-resource

True if the constraint applies to the resource as a whole; false if it only applies to one or more attributes or operations.

swarm.management.authorization-access.sensitivity-classification-constraint.types.*KEY*.classifications.*KEY*.applies-tos.*KEY*.operations

List of the names of operations to which the constraint specifically applies.

swarm.management.authorization-access.sensitivity-classification-constraint.types.*KEY*.classifications.*KEY*.configured-application

Set to override the default as to whether the constraint is considered an application resource.

swarm.management.authorization-access.sensitivity-classification-constraint.types.*KEY*.classifications.*KEY*.default-application

Whether targets having this application type constraint are considered application resources.

swarm.management.authorization-access.standard-role-names

The official names of the standard roles supported by the current management access control provider.

swarm.management.authorization-access.use-identity-roles

Should the raw roles obtained from the underlying security identity be used directly?

swarm.management.authorization-access.vault-expression-constraint.configured-requires-read

Set to override the default as to whether reading attributes containing vault expressions should be considered sensitive.

swarm.management.authorization-access.vault-expression-constraint.configured-requires-write

Set to override the default as to whether writing attributes containing vault expressions should be considered sensitive.

swarm.management.authorization-access.vault-expression-constraint.default-requires-read

Whether reading attributes containing vault expressions should be considered sensitive.

swarm.management.authorization-access.vault-expression-constraint.default-requires-write

Whether writing attributes containing vault expressions should be considered sensitive.

swarm.management.bind.interface

Interface to bind for the management ports

swarm.management.configuration-changes-service.max-history

The maximum number of configuration changes stored in history.

swarm.management.http-interface-management-interface.allowed-origins

Comma separated list of trusted Origins for sending Cross-Origin Resource Sharing requests on the management API once the user is authenticated.

swarm.management.http-interface-management-interface.console-enabled

Flag that indicates admin console is enabled

swarm.management.http-interface-management-interface.http-authentication-factory

The authentication policy to use to secure the interface for normal HTTP requests.

swarm.management.http-interface-management-interface.http-upgrade

HTTP Upgrade specific configuration

swarm.management.http-interface-management-interface.http-upgrade-enabled

Flag that indicates HTTP Upgrade is enabled, which allows HTTP requests to be upgraded to native remoting connections

swarm.management.http-interface-management-interface.sasl-protocol

The name of the protocol to be passed to the SASL mechanisms used for authentication.

swarm.management.http-interface-management-interface.secure-socket-binding

The name of the socket binding configuration to use for the HTTPS management interface's socket. When defined at least one of ssl-context or security-realm must also be defined.

swarm.management.http-interface-management-interface.security-realm

The legacy security realm to use for the HTTP management interface.

swarm.management.http-interface-management-interface.server-name

The name of the server used in the initial Remoting exchange and within the SASL mechanisms.

swarm.management.http-interface-management-interface.socket-binding

The name of the socket binding configuration to use for the HTTP management interface's socket.

swarm.management.http-interface-management-interface.ssl-context

Reference to the SSLContext to use for this management interface.

swarm.management.http.disable

Flag to disable HTTP access to management interface

swarm.management.http.port

Port for HTTP access to management interface

swarm.management.https.port

Port for HTTPS access to management interface

swarm.management.identity-access.security-domain

Reference to the security domain to use to obtain the current identity performing a management request.

swarm.management.ldap-connections.*KEY*.always-send-client-cert

If true, the client SSL certificate will be sent to LDAP server with every request; otherwise the client SSL certificate will not be sent when verifying the user credentials

swarm.management.ldap-connections.*KEY*.handles-referrals-for

List of URLs that this connection handles referrals for.

swarm.management.ldap-connections.*KEY*.initial-context-factory

The initial context factory to establish the LdapContext.

swarm.management.ldap-connections.*KEY*.properties.*KEY*.value

The optional value of the property.

swarm.management.ldap-connections.*KEY*.referrals

The referral handling mode for this connection.

swarm.management.ldap-connections.*KEY*.search-credential

The credential to use when connecting to perform a search.

swarm.management.ldap-connections.*KEY*.search-credential-reference

The reference to the search credential stored in CredentialStore under defined alias or clear text password.

swarm.management.ldap-connections.*KEY*.search-dn

The distinguished name to use when connecting to the LDAP server to perform searches.

swarm.management.ldap-connections.*KEY*.security-realm

The security realm to reference to obtain a configured SSLContext to use when establishing the connection.

swarm.management.ldap-connections.*KEY*.url

The URL to use to connect to the LDAP server.

swarm.management.management-operations-service.active-operations.*KEY*.access-mechanism

The mechanism used to submit a request to the server.

swarm.management.management-operations-service.active-operations.*KEY*.address

The address of the resource targeted by the operation. The value in the final element of the address will be '<hidden>' if the caller is not authorized to address the operation's target resource.

swarm.management.management-operations-service.active-operations.*KEY*.caller-thread

The name of the thread that is executing the operation.

swarm.management.management-operations-service.active-operations.*KEY*.cancelled

Whether the operation has been cancelled.

swarm.management.management-operations-service.active-operations.*KEY*.domain-rollout

True if the operation is a subsidiary request on a domain process other than the one directly handling the original operation, executing locally as part of the rollout of the original operation across the domain.

swarm.management.management-operations-service.active-operations.*KEY*.domain-uuid

Identifier of an overall multi-process domain operation of which this operation is a part, or undefined if this operation is not associated with such a domain operation.

swarm.management.management-operations-service.active-operations.*KEY*.exclusive-running-time

Amount of time the operation has been executing with the exclusive operation execution lock held, or -1 if the operation does not hold the exclusive execution lock.

swarm.management.management-operations-service.active-operations.*KEY*.execution-status

The current activity of the operation.

swarm.management.management-operations-service.active-operations.*KEY*.operation

The name of the operation, or '<hidden>' if the caller is not authorized to address the operation's target resource.

swarm.management.management-operations-service.active-operations.*KEY*.running-time

Amount of time the operation has been executing.

swarm.management.native-interface-management-interface.sasl-authentication-factory

The SASL authentication policy to use to secure this interface.

swarm.management.native-interface-management-interface.sasl-protocol

The name of the protocol to be passed to the SASL mechanisms used for authentication.

swarm.management.native-interface-management-interface.security-realm

The legacy security realm to use for the native management interface.

swarm.management.native-interface-management-interface.server-name

The name of the server used in the initial Remoting exchange and within the SASL mechanisms.

swarm.management.native-interface-management-interface.socket-binding

The name of the socket binding configuration to use for the native management interface's socket.

swarm.management.native-interface-management-interface.ssl-context

Reference to the SSLContext to use for this management interface.

swarm.management.security-realms.*KEY*.jaas-authentication.assign-groups

Map the roles loaded by JAAS to groups.

swarm.management.security-realms.*KEY*.jaas-authentication.name

The name of the JAAS configuration to use.

swarm.management.security-realms.*KEY*.kerberos-authentication.remove-realm

After authentication should the realm name be stripped from the users name.

swarm.management.security-realms.*KEY*.kerberos-server-identity.keytabs.*KEY*.debug

Should additional debug logging be enabled during TGT acquisition?

swarm.management.security-realms.*KEY*.kerberos-server-identity.keytabs.*KEY*.for-hosts

A server can be accessed using different host names, this attribute specifies which host names this keytab can be used with.

swarm.management.security-realms.*KEY*.kerberos-server-identity.keytabs.*KEY*.path

The path to the keytab.

swarm.management.security-realms.*KEY*.kerberos-server-identity.keytabs.*KEY*.relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.management.security-realms.*KEY*.ldap-authentication.advanced-filter

The fully defined filter to be used to search for the user based on their entered user ID. The filter should contain a variable in the form {0} - this will be replaced with the username supplied by the user.

swarm.management.security-realms.*KEY*.ldap-authentication.allow-empty-passwords

Should empty passwords be accepted from the user being authenticated.

swarm.management.security-realms.*KEY*.ldap-authentication.base-dn

The base distinguished name to commence the search for the user.

swarm.management.security-realms.*KEY*.ldap-authentication.by-access-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authentication.by-access-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authentication.by-access-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authentication.by-access-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authentication.by-search-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authentication.by-search-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authentication.by-search-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authentication.by-search-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authentication.connection

The name of the connection to use to connect to LDAP.

swarm.management.security-realms.*KEY*.ldap-authentication.recursive

Whether the search should be recursive.

swarm.management.security-realms.*KEY*.ldap-authentication.user-dn

The name of the attribute which is the user's distinguished name.

swarm.management.security-realms.*KEY*.ldap-authentication.username-attribute

The name of the attribute to search for the user. This filter will then perform a simple search where the username entered by the user matches the attribute specified here.

swarm.management.security-realms.*KEY*.ldap-authentication.username-load

The name of the attribute that should be loaded from the authenticated users LDAP entry to replace the username that they supplied, e.g. convert an e-mail address to an ID or correct the case entered.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.base-dn

The starting point of the search for the user.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-access-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-access-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-access-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-access-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-search-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-search-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-search-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-search-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.filter

The filter to use for the LDAP search.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.force

Authentication may have already converted the username to a distinguished name, force this to occur again before loading groups.

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.recursive

Should levels below the starting point be recursively searched?

swarm.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.user-dn-attribute

The attribute on the user entry that contains their distinguished name.

swarm.management.security-realms.*KEY*.ldap-authorization.connection

The name of the connection to use to connect to LDAP.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.base-dn

The starting point of the search for the group.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-access-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-access-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-access-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-access-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-search-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-search-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-search-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-search-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.group-dn-attribute

Which attribute on a group entry is it's distinguished name.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.group-name

An enumeration to identify if groups should be referenced using a simple name or their distinguished name.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.group-name-attribute

Which attribute on a group entry is it's simple name.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.iterative

Should further searches be performed to identify groups that the groups identified are a member of?

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.prefer-original-connection

After following a referral should subsequent searches prefer the original connection or use the connection of the last referral.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.principal-attribute

The attribute on the group entry that references the principal.

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.recursive

Should levels below the starting point be recursively searched?

swarm.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.search-by

Should searches be performed using simple names or distinguished names?

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-access-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-access-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-access-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-access-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-search-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-search-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-search-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-search-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.group-attribute

The attribute on the principal which references the group the principal is a member of.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.group-dn-attribute

Which attribute on a group entry is it's distinguished name.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.group-name

An enumeration to identify if groups should be referenced using a simple name or their distinguished name.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.group-name-attribute

Which attribute on a group entry is it's simple name.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.iterative

Should further searches be performed to identify groups that the groups identified are a member of?

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.parse-group-name-from-dn

Should the group name be extracted from the distinguished name.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.prefer-original-connection

After following a referral should subsequent searches prefer the original connection or use the connection of the last referral.

swarm.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.skip-missing-groups

If a non-existent group is referenced should it be quietly ignored.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.attribute

The attribute on the user entry that is their username.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.base-dn

The starting point of the search for the user.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-access-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-access-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-access-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-access-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-search-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-search-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-search-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-search-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.force

Authentication may have already converted the username to a distinguished name, force this to occur again before loading groups.

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.recursive

Should levels below the starting point be recursively searched?

swarm.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.user-dn-attribute

The attribute on the user entry that contains their distinguished name.

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-access-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-access-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-access-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-access-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-search-time-cache.cache-failures

Should failures be cached?

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-search-time-cache.cache-size

The current size of the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-search-time-cache.eviction-time

The time in seconds until an entry should be evicted from the cache.

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-search-time-cache.max-cache-size

The maximum size of the cache before the oldest items are removed to make room for new entries.

swarm.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.force

Authentication may have already converted the username to a distinguished name, force this to occur again before loading groups.

swarm.management.security-realms.*KEY*.local-authentication.allowed-users

The comma separated list of users that will be accepted using the JBOSS-LOCAL-USER mechanism or '*' to accept all. If specified the default-user is always assumed allowed.

swarm.management.security-realms.*KEY*.local-authentication.default-user

The name of the default user to assume if no user specified by the remote client.

swarm.management.security-realms.*KEY*.local-authentication.skip-group-loading

Disable the loading of the users group membership information after local authentication has been used.

swarm.management.security-realms.*KEY*.map-groups-to-roles

After a users group membership has been loaded should a 1:1 relationship be assumed regarding group to role mapping.

swarm.management.security-realms.*KEY*.plug-in-authentication.mechanism

Allow the mechanism this plug-in is compatible with to be overridden from DIGEST.

swarm.management.security-realms.*KEY*.plug-in-authentication.name

The short name of the plug-in (as registered) to use.

swarm.management.security-realms.*KEY*.plug-in-authentication.properties.*KEY*.value

The optional value of the property.

swarm.management.security-realms.*KEY*.plug-in-authorization.name

The short name of the plug-in (as registered) to use.

swarm.management.security-realms.*KEY*.plug-in-authorization.properties.*KEY*.value

The optional value of the property.

swarm.management.security-realms.*KEY*.properties-authentication.path

The path of the properties file containing the users.

swarm.management.security-realms.*KEY*.properties-authentication.plain-text

Are the credentials within the properties file stored in plain text. If not the credential is expected to be the hex encoded Digest hash of 'username : realm : password'.

swarm.management.security-realms.*KEY*.properties-authentication.relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.management.security-realms.*KEY*.properties-authorization.path

The path of the properties file containing the users roles.

swarm.management.security-realms.*KEY*.properties-authorization.relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.management.security-realms.*KEY*.secret-server-identity.credential-reference

The reference to credential for the secret / password stored in CredentialStore under defined alias or clear text password.

swarm.management.security-realms.*KEY*.secret-server-identity.value

The secret / password - Base64 Encoded.

swarm.management.security-realms.*KEY*.ssl-server-identity.alias

The alias of the entry to use from the keystore.

swarm.management.security-realms.*KEY*.ssl-server-identity.enabled-cipher-suites

The cipher suites that can be enabled on the underlying SSLEngine.

swarm.management.security-realms.*KEY*.ssl-server-identity.enabled-protocols

The protocols that can be enabled on the underlying SSLEngine.

swarm.management.security-realms.*KEY*.ssl-server-identity.generate-self-signed-certificate-host

If the keystore does not exist and this attribute is set then a self signed certificate will be generated for the specified host name. This is not intended for production use.

swarm.management.security-realms.*KEY*.ssl-server-identity.key-password

The password to obtain the key from the keystore.

swarm.management.security-realms.*KEY*.ssl-server-identity.key-password-credential-reference

The reference to credential for the keystore key stored in CredentialStore under defined alias or clear text password.

swarm.management.security-realms.*KEY*.ssl-server-identity.keystore-password

The password to open the keystore.

swarm.management.security-realms.*KEY*.ssl-server-identity.keystore-password-credential-reference

The reference to credential for the keystore password stored in CredentialStore under defined alias or clear text password.

swarm.management.security-realms.*KEY*.ssl-server-identity.keystore-path

The path of the keystore, will be ignored if the keystore-provider is anything other than JKS.

swarm.management.security-realms.*KEY*.ssl-server-identity.keystore-provider

The provider for loading the keystore, defaults to JKS.

swarm.management.security-realms.*KEY*.ssl-server-identity.keystore-relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.management.security-realms.*KEY*.ssl-server-identity.protocol

The protocol to use when creating the SSLContext.

swarm.management.security-realms.*KEY*.truststore-authentication.keystore-password

The password to open the keystore.

swarm.management.security-realms.*KEY*.truststore-authentication.keystore-password-credential-reference

The reference to credential for the keystore password stored in CredentialStore under defined alias or clear text password.

swarm.management.security-realms.*KEY*.truststore-authentication.keystore-path

The path of the keystore, will be ignored if the keystore-provider is anything other than JKS.

swarm.management.security-realms.*KEY*.truststore-authentication.keystore-provider

The provider for loading the keystore, defaults to JKS.

swarm.management.security-realms.*KEY*.truststore-authentication.keystore-relative-to

The name of another previously named path, or of one of the standard paths provided by the system. If 'relative-to' is provided, the value of the 'path' attribute is treated as relative to the path specified by this attribute.

swarm.management.security-realms.*KEY*.users-authentication.users.*KEY*.credential-reference

The reference to credential for the password stored in CredentialStore under defined alias or clear text password.

swarm.management.security-realms.*KEY*.users-authentication.users.*KEY*.password

The user's password.

D.23. MICROPROFILE

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile</artifactId>
</dependency>
```

D.23.1. MicroProfile Config

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-config</artifactId>
</dependency>
```

Configuration

swarm.microprofile.config.config-source-providers.*KEY*.attribute-class

Class of the ConfigSourceProvider to load

swarm.microprofile.config.config-sources.*KEY*.attribute-class

Class of the config source to load

swarm.microprofile.config.config-sources.*KEY*.dir

Directory that is scanned to config properties for this config source (file names are key, file content are value)

swarm.microprofile.config.config-sources.*KEY*.ordinal

Ordinal value for the config source

swarm.microprofile.config.config-sources.*KEY*.properties

Properties configured for this config source

D.23.2. MicroProfile Fault Tolerance

This fraction implements the Eclipse MicroProfile Fault Tolerance API. The implementation depends on the Hystrix fraction, which is added transitively into your application. Use standard configuration mechanisms to configure Hystrix properties in your application.

D.23.2.1. Bulkhead fallback rejection

If you use the **`@Bulkhead`** pattern together with some **`@Fallback`** logic to limit the number of concurrent requests, an invocation may still result in an exception.

D.23.2.1.1. Semaphore Isolation

For semaphore-style **`@Bulkhead`** a **`BulkheadException`** may be thrown if the maximum concurrent limit is reached. To avoid that, set the **`swarm.hystrix.command.default.fallback.isolation.semaphore.maxConcurrentRequests`** property to increase the limit.

D.23.2.1.2. Thread Isolation

For **`@Bulkhead`** used together with **`@Asynchronous`** a **`RejectedExecutionException`** may be thrown if the maximum concurrent limit is reached. To avoid that, set the **`swarm.hystrix.threadpool.default.maximumSize`** property to increase the limit. Also don't forget to set the **`swarm.hystrix.threadpool.default.allowMaximumSizeToDivergeFromCoreSize`** property to **`true`**.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-fault-tolerance</artifactId>
</dependency>
```

Configuration

swarm.microprofile.fault-tolerance.synchronous-circuit-breaker

Enable/disable synchronous circuit breaker functionality. If disabled, **`CircuitBreaker#successThreshold()`** of value greater than 1 is not supported. Moreover, circuit breaker does not necessarily transition from **`CLOSED`** to **`OPEN`** immediately when a fault tolerance operation completes. However, applications are encouraged to disable this feature on high-volume circuits.

D.23.3. MicroProfile Health

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-health</artifactId>
</dependency>
```

Configuration

swarm.microprofile.health.security-realm

Security realm configuration

D.23.4. MicroProfile JWT RBAC Auth

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-jwt</artifactId>
</dependency>
```

Configuration

swarm.microprofile.jwt.default-missing-method-permissions-deny-access

If a JAX-RS resource has no class-level security metadata, then if this property is set to **`true`** and at least one resource method has security metadata all other resource methods without security metadata have an implicit **`@DenyAll`**, otherwise resource methods without security metadata are not secured

swarm.microprofile.jwt.token.exp-grace-period

The JWT token expiration grace period in seconds

swarm.microprofile.jwt.token.issued-by

The URI of the JWT token issuer

swarm.microprofile.jwt.token.jwks-refresh-interval

The interval at which the JWKS URI should be queried for keys (in minutes).

swarm.microprofile.jwt.token.jwks-uri

The JWKS URI from which to load public keys (if 'signer-pub-key' is set, this setting is ignored).

swarm.microprofile.jwt.token.signer-pub-key

The public key of the JWT token signer. Can be prefixed 'file:' or 'classpath:' for key assets, otherwise the key contents are expected

D.23.5. MicroProfile Metrics

This fraction implements the MicroProfile Metrics 1.0 specification.

To use this in your project you need the following in your pom.xml

```
<dependency>
   <groupId>io.thorntail</groupId>
   <artifactId>microprofile-metrics</artifactId>
 </dependency>
```

There is no need to include the MicroProfile Metrics API dependency, as it comes with the fraction.

By default the base metrics and vendor metrics of the server are exposed as required by the spec.

NOTE

Exposing application metrics currently only works if you chose **`war`** packaging of your application

```
<project>
  <groupId>org.example</groupId>
  <artifactId>thorntail-demo</artifactId>
  <packaging>war</packaging> (1)
```

(1) war packaging

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-metrics</artifactId>
</dependency>
```

D.23.6. MicroProfile OpenAPI

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-openapi</artifactId>
</dependency>
```

D.23.7. MicroProfile OpenTracing

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-opentracing</artifactId>
</dependency>
```

D.23.8. MicroProfile Rest Client

D.23.8.1. CDI Interceptors Support

In general, Rest Client proxies are not created by the CDI container and therefore method invocations do not pass through CDI interceptors. In Thorntail, however, you can associate business method interceptors (denoted by the **`@AroundInvoke`** annotation) with a Rest Client proxy by using interceptor bindings. This feature is non-portable. The primary use case is the support of Section D.23.2, “MicroProfile Fault Tolerance” annotations, for example:

```
import org.eclipse.microprofile.faulttolerance.Retry;

@Path("/v1")
```

```
interface MyClient {

    @Retry(maxRetries = 3) // Retry on any exception thrown
    @GET
    @Path("/hello")
    String hello();
}
```

NOTE

The **`org.eclipse.microprofile.faulttolerance.Asynchronous`** annotation is currently not supported because the underlying RESTEasy client is not able to handle the **`java.util.concurrent.Future`** return types.

D.23.8.2. RestClientProxy

In addition to the MicroProfile Rest Client specification, every Rest Client proxy implements **`io.smallrye.restclient.RestClientProxy`** interface which allows you to:

- obtain the underlying **`javax.ws.rs.client.Client`** instance
- release all associated resources, for example:

```
public void hello() {
   MyClient myClient =
RestClientBuilder.newBuilder().build(MyClient.class);
   myClient.hello();
   // Finally release all associated resources
   ((RestClientProxy) helloClient).close();
}
```

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-restclient</artifactId>
</dependency>
```

D.24. MONITOR

WARNING

This fraction is deprecated. Use the **`io.thorntail:microprofile-health`** fraction instead.

Maven Coordinates

```
<dependency>
```

```
  <groupId>io.thorntail</groupId>
  <artifactId>monitor</artifactId>
</dependency>
```

Configuration

swarm.monitor.security-realm

(not yet documented)

D.25. MSC

Primarily an internal fraction providing support for the JBoss Modular Container (MSC). JBoss MSC provides the underpinning for all services wired together supporting the container and the application.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>msc</artifactId>
</dependency>
```

D.26. NAMING

Provides support for JNDI.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>naming</artifactId>
</dependency>
```

Configuration

swarm.naming.bindings.*KEY*.attribute-class

The object factory class name for object factory bindings

swarm.naming.bindings.*KEY*.binding-type

The type of binding to create, may be simple, lookup, external-context or object-factory

swarm.naming.bindings.*KEY*.cache

If the external context should be cached

swarm.naming.bindings.*KEY*.environment

The environment to use on object factory instance retrieval

swarm.naming.bindings.*KEY*.lookup

The entry to lookup in JNDI for lookup bindings

swarm.naming.bindings.*KEY*.module

The module to load the object factory from for object factory bindings

swarm.naming.bindings.*KEY*.type

The type of the value to bind for simple bindings, this must be a primitive type

swarm.naming.bindings.*KEY*.value

The value to bind for simple bindings

D.27. GUAVA

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>netflix-guava</artifactId>
</dependency>
```

D.28. RX-JAVA

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>netflix-rxjava</artifactId>
</dependency>
```

D.29. OPENTRACING

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>opentracing</artifactId>
</dependency>
```

Configuration

swarm.opentracing.servlet.skipPattern

The servlet skip pattern as a Java compilable Pattern. Optional. Ex.: **`/health-check`**

D.29.1. OpenTracing TracerResolver

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>opentracing-tracerresolver</artifactId>
</dependency>
```

D.30. REMOTING

Primarily an internal fraction providing remote invocation support for higher-level fractions such as EJB.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>remoting</artifactId>
</dependency>
```

Configuration

swarm.remoting.connectors.*KEY*.authentication-provider

The "authentication-provider" element contains the name of the authentication provider to use for incoming connections.

swarm.remoting.connectors.*KEY*.properties.*KEY*.value

The property value.

swarm.remoting.connectors.*KEY*.sasl-authentication-factory

Reference to the SASL authentication factory to secure this connector.

swarm.remoting.connectors.*KEY*.sasl-protocol

The protocol to pass into the SASL mechanisms used for authentication.

swarm.remoting.connectors.*KEY*.sasl-security.include-mechanisms

The optional nested "include-mechanisms" element contains a whitelist of allowed SASL mechanism names. No mechanisms will be allowed which are not present in this list.

swarm.remoting.connectors.*KEY*.sasl-security.policy-sasl-policy.forward-secrecy

The optional nested "forward-secrecy" element contains a boolean value which specifies whether mechanisms that implement forward secrecy between sessions are required. Forward secrecy means that breaking into one session will not automatically provide information for breaking into future sessions.

swarm.remoting.connectors.*KEY*.sasl-security.policy-sasl-policy.no-active

The optional nested "no-active" element contains a boolean value which specifies whether mechanisms susceptible to active (non-dictionary) attacks are not permitted. "false" to permit, "true" to deny.

swarm.remoting.connectors.*KEY*.sasl-security.policy-sasl-policy.no-anonymous

The optional nested "no-anonymous" element contains a boolean value which specifies whether mechanisms that accept anonymous login are permitted. "false" to permit, "true" to deny.

swarm.remoting.connectors.*KEY*.sasl-security.policy-sasl-policy.no-dictionary

The optional nested "no-dictionary" element contains a boolean value which specifies whether mechanisms susceptible to passive dictionary attacks are permitted. "false" to permit, "true" to deny.

swarm.remoting.connectors.*KEY*.sasl-security.policy-sasl-policy.no-plain-text

The optional nested "no-plain-text" element contains a boolean value which specifies whether mechanisms susceptible to simple plain passive attacks (e.g., "PLAIN") are not permitted. "false" to permit, "true" to deny.

swarm.remoting.connectors.*KEY*.sasl-security.policy-sasl-policy.pass-credentials

The optional nested "pass-credentials" element contains a boolean value which specifies whether mechanisms that pass client credentials are required.

swarm.remoting.connectors.*KEY*.sasl-security.properties.*KEY*.value

The property value.

swarm.remoting.connectors.*KEY*.sasl-security.qop

The optional nested "qop" element contains a list of quality-of-protection values, in decreasing order of preference.

swarm.remoting.connectors.*KEY*.sasl-security.reuse-session

The optional nested "reuse-session" boolean element specifies whether or not the server should attempt to reuse previously authenticated session information. The mechanism may or may not support such reuse, and other factors may also prevent it.

swarm.remoting.connectors.*KEY*.sasl-security.server-auth

The optional nested "server-auth" boolean element specifies whether the server should authenticate to the client. Not all mechanisms may support this setting.

swarm.remoting.connectors.*KEY*.sasl-security.strength

The optional nested "strength" element contains a list of cipher strength values, in decreasing order of preference.

swarm.remoting.connectors.*KEY*.security-realm

The associated security realm to use for authentication for this connector.

swarm.remoting.connectors.*KEY*.server-name

The server name to send in the initial message exchange and for SASL based authentication.

swarm.remoting.connectors.*KEY*.socket-binding

The name of the socket binding to attach to.

swarm.remoting.connectors.*KEY*.ssl-context

Reference to the SSLContext to use for this connector.

swarm.remoting.endpoint-configuration.auth-realm

The authentication realm to use if no authentication {@code CallbackHandler} is specified.

swarm.remoting.endpoint-configuration.authentication-retries

Specify the number of times a client is allowed to retry authentication before closing the connection.

swarm.remoting.endpoint-configuration.authorize-id

The SASL authorization ID. Used as authentication user name to use if no authentication {@code CallbackHandler} is specifiedand the selected SASL mechanism demands a user name.

swarm.remoting.endpoint-configuration.buffer-region-size

The size of allocated buffer regions.

swarm.remoting.endpoint-configuration.heartbeat-interval

The interval to use for connection heartbeat, in milliseconds. If the connection is idle in the outbound directionfor this amount of time, a ping message will be sent, which will trigger a corresponding reply message.

swarm.remoting.endpoint-configuration.max-inbound-channels

The maximum number of inbound channels to support for a connection.

swarm.remoting.endpoint-configuration.max-inbound-message-size

The maximum inbound message size to be allowed. Messages exceeding this size will cause an exception to be thrown on the reading side as well as the writing side.

swarm.remoting.endpoint-configuration.max-inbound-messages

The maximum number of concurrent inbound messages on a channel.

swarm.remoting.endpoint-configuration.max-outbound-channels

The maximum number of outbound channels to support for a connection.

swarm.remoting.endpoint-configuration.max-outbound-message-size

The maximum outbound message size to send. No messages larger than this well be transmitted; attempting to do so will cause an exception on the writing side.

swarm.remoting.endpoint-configuration.max-outbound-messages

The maximum number of concurrent outbound messages on a channel.

swarm.remoting.endpoint-configuration.receive-buffer-size

The size of the largest buffer that this endpoint will accept over a connection.

swarm.remoting.endpoint-configuration.receive-window-size

The maximum window size of the receive direction for connection channels, in bytes.

swarm.remoting.endpoint-configuration.sasl-protocol

Where a SaslServer or SaslClient are created by default the protocol specified it 'remoting', this can be used to override this.

swarm.remoting.endpoint-configuration.send-buffer-size

The size of the largest buffer that this endpoint will transmit over a connection.

swarm.remoting.endpoint-configuration.server-name

The server side of the connection passes it's name to the client in the initial greeting, by default the name is automatically discovered from the local address of the connection or it can be overridden using this.

swarm.remoting.endpoint-configuration.transmit-window-size

The maximum window size of the transmit direction for connection channels, in bytes.

swarm.remoting.endpoint-configuration.worker

Worker to use

swarm.remoting.http-connectors.*KEY*.authentication-provider

The "authentication-provider" element contains the name of the authentication provider to use for incoming connections.

swarm.remoting.http-connectors.*KEY*.connector-ref

The name (or names) of a connector in the Undertow subsystem to connect to.

swarm.remoting.http-connectors.*KEY*.properties.*KEY*.value

The property value.

swarm.remoting.http-connectors.*KEY*.sasl-authentication-factory

Reference to the SASL authentication factory to use for this connector.

swarm.remoting.http-connectors.*KEY*.sasl-protocol

The protocol to pass into the SASL mechanisms used for authentication.

swarm.remoting.http-connectors.*KEY*.sasl-security.include-mechanisms

The optional nested "include-mechanisms" element contains a whitelist of allowed SASL mechanism names. No mechanisms will be allowed which are not present in this list.

swarm.remoting.http-connectors.*KEY*.sasl-security.policy-sasl-policy.forward-secrecy

The optional nested "forward-secrecy" element contains a boolean value which specifies whether mechanisms that implement forward secrecy between sessions are required. Forward secrecy means that breaking into one session will not automatically provide information for breaking into future sessions.

swarm.remoting.http-connectors.*KEY*.sasl-security.policy-sasl-policy.no-active

The optional nested "no-active" element contains a boolean value which specifies whether mechanisms susceptible to active (non-dictionary) attacks are not permitted. "false" to permit, "true" to deny.

swarm.remoting.http-connectors.*KEY*.sasl-security.policy-sasl-policy.no-anonymous

The optional nested "no-anonymous" element contains a boolean value which specifies whether mechanisms that accept anonymous login are permitted. "false" to permit, "true" to deny.

swarm.remoting.http-connectors.*KEY*.sasl-security.policy-sasl-policy.no-dictionary

The optional nested "no-dictionary" element contains a boolean value which specifies whether mechanisms susceptible to passive dictionary attacks are permitted. "false" to permit, "true" to deny.

swarm.remoting.http-connectors.*KEY*.sasl-security.policy-sasl-policy.no-plain-text

The optional nested "no-plain-text" element contains a boolean value which specifies whether mechanisms susceptible to simple plain passive attacks (e.g., "PLAIN") are not permitted. "false" to permit, "true" to deny.

swarm.remoting.http-connectors.*KEY*.sasl-security.policy-sasl-policy.pass-credentials

The optional nested "pass-credentials" element contains a boolean value which specifies whether mechanisms that pass client credentials are required.

swarm.remoting.http-connectors.*KEY*.sasl-security.properties.*KEY*.value

The property value.

swarm.remoting.http-connectors.*KEY*.sasl-security.qop

The optional nested "qop" element contains a list of quality-of-protection values, in decreasing order of preference.

swarm.remoting.http-connectors.*KEY*.sasl-security.reuse-session

The optional nested "reuse-session" boolean element specifies whether or not the server should attempt to reuse previously authenticated session information. The mechanism may or may not support such reuse, and other factors may also prevent it.

swarm.remoting.http-connectors.*KEY*.sasl-security.server-auth

The optional nested "server-auth" boolean element specifies whether the server should authenticate to the client. Not all mechanisms may support this setting.

swarm.remoting.http-connectors.*KEY*.sasl-security.strength

The optional nested "strength" element contains a list of cipher strength values, in decreasing order of preference.

swarm.remoting.http-connectors.*KEY*.security-realm

The associated security realm to use for authentication for this connector.

swarm.remoting.http-connectors.*KEY*.server-name

The server name to send in the initial message exchange and for SASL based authentication.

swarm.remoting.local-outbound-connections.*KEY*.outbound-socket-binding-ref

Name of the outbound-socket-binding which will be used to determine the destination address and port for the connection.

swarm.remoting.local-outbound-connections.*KEY*.properties.*KEY*.value

The property value.

swarm.remoting.outbound-connections.*KEY*.properties.*KEY*.value

The property value.

swarm.remoting.outbound-connections.*KEY*.uri

The connection URI for the outbound connection.

swarm.remoting.port

Port for legacy remoting connector

swarm.remoting.remote-outbound-connections.*KEY*.authentication-context

Reference to the authentication context instance containing the configuration for outbound connections.

swarm.remoting.remote-outbound-connections.*KEY*.outbound-socket-binding-ref

Name of the outbound-socket-binding which will be used to determine the destination address and port for the connection.

swarm.remoting.remote-outbound-connections.*KEY*.properties.*KEY*.value

The property value.

swarm.remoting.remote-outbound-connections.*KEY*.protocol

The protocol to use for the remote connection.

swarm.remoting.remote-outbound-connections.*KEY*.security-realm

Reference to the security realm to use to obtain the password and SSL configuration.

swarm.remoting.remote-outbound-connections.*KEY*.username

The user name to use when authenticating against the remote server.

swarm.remoting.required

(not yet documented)

swarm.remoting.worker-read-threads

The number of read threads to create for the remoting worker.

swarm.remoting.worker-task-core-threads

The number of core threads for the remoting worker task thread pool.

swarm.remoting.worker-task-keepalive

The number of milliseconds to keep non-core remoting worker task threads alive.

swarm.remoting.worker-task-limit

The maximum number of remoting worker tasks to allow before rejecting.

swarm.remoting.worker-task-max-threads

The maximum number of threads for the remoting worker task thread pool.

swarm.remoting.worker-write-threads

The number of write threads to create for the remoting worker.

D.31. REQUEST CONTROLLER

Provides support for the JBoss EAP request-controller, allowing for graceful pause/resume/shutdown of the container.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>request-controller</artifactId>
</dependency>
```

Configuration

swarm.request-controller.active-requests

The number of requests that are currently running in the server

swarm.request-controller.max-requests

The maximum number of all types of requests that can be running in a server at a time. Once this limit is hit any new requests will be rejected.

swarm.request-controller.track-individual-endpoints

If this is true requests are tracked at an endpoint level, which will allow individual deployments to be suspended

D.32. RESOURCE ADAPTERS

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>resource-adapters</artifactId>
</dependency>
```

Configuration

swarm.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.class-name

Specifies the fully qualified class name of an administration object.

swarm.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.config-properties.*KEY*.value

Custom defined config property value.

swarm.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.enabled

Specifies if the administration object should be enabled.

swarm.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.jndi-name

Specifies the JNDI name for the administration object.

swarm.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.use-java-context

Setting this to false will bind the object into global JNDI.

swarm.resource-adapters.resource-adapters.*KEY*.archive

Specifies the resource adapter archive.

swarm.resource-adapters.resource-adapters.*KEY*.beanvalidationgroups

Specifies the bean validation groups that should be used.

swarm.resource-adapters.resource-adapters.*KEY*.bootstrap-context

Specifies the unique name of the bootstrap context that should be used.

swarm.resource-adapters.resource-adapters.*KEY*.config-properties.*KEY*.value

Custom defined config property value.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.allocation-retry

The allocation retry element indicates the number of times that allocating a connection should be tried before throwing an exception.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.allocation-retry-wait-millis

The allocation retry wait millis element specifies the amount of time, in milliseconds, to wait between retrying to allocate a connection.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.authentication-context

The Elytron authentication context which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.authentication-context-and-application

Indicates that either application-supplied parameters, such as from getConnection(user, pw), or Subject (provided by Elytron after authenticating using configured authentication-context), are used to distinguish connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.background-validation

An element to specify that connections should be validated on a background thread versus being validated prior to use. Changing this value requires a server restart.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.background-validation-millis

The background-validation-millis element specifies the amount of time, in milliseconds, that background validation will run. Changing this value requires a server restart.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.blocking-timeout-wait-millis

The blocking-timeout-millis element specifies the maximum time, in milliseconds, to block while waiting for a connection before throwing an exception. Note that this blocks only while waiting for locking a connection, and will never throw an exception if creating a new connection takes an inordinately long time.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.capacity-decrementer-class

Class defining the policy for decrementing connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.capacity-decrementer-properties

Properties to inject in class defining the policy for decrementing connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.capacity-incrementer-class

Class defining the policy for incrementing connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.capacity-incrementer-properties

Properties to inject in class defining the policy for incrementing connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.class-name

Specifies the fully qualified class name of a managed connection factory or admin object.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.config-properties.*KEY*.value

Custom defined config property value.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.connectable

Enable the use of CMR. This feature means that a local resource can reliably participate in an XA transaction.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.elytron-enabled

Enables Elytron security for handling authentication of connections. The Elytron authentication-context to be used will be current context if no context is specified (see authentication-context).

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.enabled

Specifies if the resource adapter should be enabled.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.enlistment

Defines if lazy enlistment should be used if supported by the resource adapter.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.enlistment-trace

Defines if WildFly/IronJacamar should record enlistment traces.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.flush-strategy

Specifies how the pool should be flushed in case of an error.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.idle-timeout-minutes

Specifies the maximum time, in minutes, a connection may be idle before being closed. The actual maximum time depends also on the IdleRemover scan time, which is half of the smallest idle-timeout-minutes value of any pool. Changing this value requires a server restart.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.initial-pool-size

Specifies the initial number of connections a pool should hold.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.interleaving

An element to enable interleaving for XA connections.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.jndi-name

Specifies the JNDI name for the connection factory.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.max-pool-size

Specifies the maximum number of connections for a pool. No more connections will be created in each sub-pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.mcp

Defines the ManagedConnectionPool implementation. For example: org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.min-pool-size

Specifies the minimum number of connections for a pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.no-recovery

Specifies if the connection pool should be excluded from recovery.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.no-tx-separate-pool

Oracle does not like XA connections getting used both inside and outside a JTA transaction. To workaround the problem you can create separate sub-pools for the different contexts.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.pad-xid

Specifies whether the Xid should be padded.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.pool-fair

Defines if pool use should be fair.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.pool-prefill

Specifies if the pool should be prefilled. Changing this value requires a server restart.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.pool-use-strict-min

Specifies if the min-pool-size should be considered strict.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-authentication-context

The Elytron authentication context used for recovery (current authentication-context will be used if unspecified).

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-credential-reference

Credential (from Credential Store) to authenticate on recovery connection

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-elytron-enabled

Indicates that an Elytron authentication context will be used for recovery.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-password

The password used for recovery.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-plugin-class-name

The fully qualified class name of the recovery plugin implementation.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-plugin-properties

The properties for the recovery plugin.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-security-domain

The PicketBox security domain used for recovery.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-username

The user name used for recovery.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.same-rm-override

Using this attribute, you can unconditionally set whether javax.transaction.xa.XAResource.isSameRM(XAResource) returns true or false.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.security-application

Indicates that application-supplied parameters, such as from getConnection(user, pw), are used to distinguish connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.security-domain

Specifies the PicketBox security domain which defines the javax.security.auth.Subject that is used to distinguish connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.security-domain-and-application

Indicates that either application-supplied parameters, such as from getConnection(user, pw), or Subject (from PicketBox security domain), are used to distinguish connections in the pool.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.sharable

Enable the use of sharable connections, which allows lazy association to be enabled if supported.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.tracking

Defines if IronJacamar should track connection handles across transaction boundaries.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.use-ccm

Enable the use of a cached connection manager.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.use-fast-fail

Whether to fail a connection allocation on the first try if it is invalid (true) or keep trying until the pool is exhausted of all potential connections (false).

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.use-java-context

Setting this to false will bind the object into global JNDI.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.validate-on-match

This specifies if connection validation should be done when a connection factory attempts to match a managed connection. This is typically exclusive to the use of background validation.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.wrap-xa-resource

Specifies whether XAResource instances should be wrapped in an org.jboss.tm.XAResourceWrapper instance.

swarm.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.xa-resource-timeout

The value is passed to XAResource.setTransactionTimeout(), in seconds.

swarm.resource-adapters.resource-adapters.*KEY*.module

Specifies the module from which resource adapter will be loaded

swarm.resource-adapters.resource-adapters.*KEY*.statistics-enabled

Define whether runtime statistics are enabled or not.

swarm.resource-adapters.resource-adapters.*KEY*.transaction-support

Specifies the transaction support level of the resource adapter.

swarm.resource-adapters.resource-adapters.*KEY*.wm-elytron-security-domain

Defines the name of the Elytron security domain that should be used.

swarm.resource-adapters.resource-adapters.*KEY*.wm-security

Toggle on/off wm.security for this resource adapter. In case of false all wm-security-* parameters are ignored, even the defaults.

swarm.resource-adapters.resource-adapters.*KEY*.wm-security-default-groups

Defines a default groups list that should be added to the used Subject instance.

swarm.resource-adapters.resource-adapters.*KEY*.wm-security-default-principal

Defines a default principal name that should be added to the used Subject instance.

swarm.resource-adapters.resource-adapters.*KEY*.wm-security-domain

Defines the name of the PicketBox security domain that should be used.

swarm.resource-adapters.resource-adapters.*KEY*.wm-security-mapping-groups

List of groups mappings.

swarm.resource-adapters.resource-adapters.*KEY*.wm-security-mapping-required

Defines if a mapping is required for security credentials.

swarm.resource-adapters.resource-adapters.*KEY*.wm-security-mapping-users

List of user mappings.

D.33. SECURITY

Provides underlying security infrastructure to support JAAS and other security APIs.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>security</artifactId>
</dependency>
```

Configuration

swarm.security.classic-vault.code

Fully Qualified Name of the Security Vault Implementation.

swarm.security.classic-vault.vault-options

Security Vault options.

swarm.security.deep-copy-subject-mode

Sets the copy mode of subjects done by the security managers to be deep copies that makes copies of the subject principals and credentials if they are cloneable. It should be set to true if subject include mutable content that can be corrupted when multiple threads have the same identity and cache flushes/logout clearing the subject in one thread results in subject references affecting other threads.

swarm.security.elytron-key-managers.*KEY*.legacy-jsse-config

The name of the legacy security domain that contains a JSSE configuration that can be used to export the key manager.

swarm.security.elytron-key-stores.*KEY*.legacy-jsse-config

The name of the legacy security domain that contains a JSSE configuration that can be used to export the key store.

swarm.security.elytron-realms.*KEY*.apply-role-mappers

Indicates to the realm if it should apply the role mappers defined in the legacy domain to the roles obtained from authenticated Subjects or not.

swarm.security.elytron-realms.*KEY*.legacy-jaas-config

The name of the legacy security domain to which authentication will be delegated.

swarm.security.elytron-trust-managers.*KEY*.legacy-jsse-config

The name of the legacy security domain that contains a JSSE configuration that can be used to export the trust manager.

swarm.security.elytron-trust-stores.*KEY*.legacy-jsse-config

The name of the legacy security domain that contains a JSSE configuration that can be used to export the trust store.

swarm.security.initialize-jacc

Indicates if this subsystem should be in charge of initializing JACC related services.

swarm.security.security-domains.*KEY*.cache-type

Adds a cache to speed up authentication checks. Allowed values are 'default' to use simple map as the cache and 'infinispan' to use an Infinispan cache.

swarm.security.security-domains.*KEY*.classic-acl.acl-modules.*KEY*.code

Class name of the module to be instantiated.

swarm.security.security-domains.*KEY*.classic-acl.acl-modules.*KEY*.flag

The flag controls how the module participates in the overall procedure. Allowed values are requisite, required, sufficient or optional.

swarm.security.security-domains.*KEY*.classic-acl.acl-modules.*KEY*.module

Name of JBoss Module where the login module is located.

swarm.security.security-domains.*KEY*.classic-acl.acl-modules.*KEY*.module-options

List of module options containing a name/value pair.

swarm.security.security-domains.*KEY*.classic-audit.provider-modules.*KEY*.code

Class name of the module to be instantiated.

swarm.security.security-domains.*KEY*.classic-audit.provider-modules.*KEY*.module

Name of JBoss Module where the mapping module code is located.

swarm.security.security-domains.*KEY*.classic-audit.provider-modules.*KEY*.module-options

List of module options containing a name/value pair.

swarm.security.security-domains.*KEY*.classic-authentication.login-modules.*KEY*.code

Class name of the module to be instantiated.

swarm.security.security-domains.*KEY*.classic-authentication.login-modules.*KEY*.flag

The flag controls how the module participates in the overall procedure. Allowed values are requisite, required, sufficient or optional.

swarm.security.security-domains.*KEY*.classic-authentication.login-modules.*KEY*.module

Name of JBoss Module where the login module is located.

swarm.security.security-domains.*KEY*.classic-authentication.login-modules.*KEY*.module-options

List of module options containing a name/value pair.

swarm.security.security-domains.*KEY*.classic-authorization.policy-modules.*KEY*.code

Class name of the module to be instantiated.

swarm.security.security-domains.*KEY*.classic-authorization.policy-modules.*KEY*.flag

The flag controls how the module participates in the overall procedure. Allowed values are requisite, required, sufficient or optional.

swarm.security.security-domains.*KEY*.classic-authorization.policy-modules.*KEY*.module

Name of JBoss Module where the login module is located.

swarm.security.security-domains.*KEY*.classic-authorization.policy-modules.*KEY*.module-options

List of module options containing a name/value pair.

swarm.security.security-domains.*KEY*.classic-identity-trust.trust-modules.*KEY*.code

Class name of the module to be instantiated.

swarm.security.security-domains.*KEY*.classic-identity-trust.trust-modules.*KEY*.flag

The flag controls how the module participates in the overall procedure. Allowed values are requisite, required, sufficient or optional.

swarm.security.security-domains.*KEY*.classic-identity-trust.trust-modules.*KEY*.module

Name of JBoss Module where the login module is located.

swarm.security.security-domains.*KEY*.classic-identity-trust.trust-modules.*KEY*.module-options

List of module options containing a name/value pair.

swarm.security.security-domains.*KEY*.classic-jsse.additional-properties

Additional properties that may be necessary to configure JSSE.

swarm.security.security-domains.*KEY*.classic-jsse.cipher-suites

Comma separated list of cipher suites to enable on SSLSockets.

swarm.security.security-domains.*KEY*.classic-jsse.client-alias

Preferred alias to use when the KeyManager chooses the client alias.

swarm.security.security-domains.*KEY*.classic-jsse.client-auth

Boolean attribute to indicate if client's certificates should also be authenticated on the server side.

swarm.security.security-domains.*KEY*.classic-jsse.key-manager

JSEE Key Manager factory

swarm.security.security-domains.*KEY*.classic-jsse.keystore

Configures a JSSE key store

swarm.security.security-domains.*KEY*.classic-jsse.protocols

Comma separated list of protocols to enable on SSLSockets.

swarm.security.security-domains.*KEY*.classic-jsse.server-alias

Preferred alias to use when the KeyManager chooses the server alias.

swarm.security.security-domains.*KEY*.classic-jsse.service-auth-token

Token to retrieve PrivateKeys from the KeyStore.

swarm.security.security-domains.*KEY*.classic-jsse.trust-manager

JSEE Trust Manager factory

swarm.security.security-domains.*KEY*.classic-jsse.truststore

Configures a JSSE trust store

swarm.security.security-domains.*KEY*.classic-mapping.mapping-modules.*KEY*.code

Class name of the module to be instantiated.

swarm.security.security-domains.*KEY*.classic-mapping.mapping-modules.*KEY*.module

Name of JBoss Module where the mapping module code is located.

swarm.security.security-domains.*KEY*.classic-mapping.mapping-modules.*KEY*.module-options

List of module options containing a name/value pair.

swarm.security.security-domains.*KEY*.classic-mapping.mapping-modules.*KEY*.type

Type of mapping this module performs. Allowed values are principal, role, attribute or credential..

swarm.security.security-domains.*KEY*.jaspi-authentication.auth-modules.*KEY*.code

Class name of the module to be instantiated.

swarm.security.security-domains.*KEY*.jaspi-authentication.auth-modules.*KEY*.flag

The flag controls how the module participates in the overall procedure. Allowed values are requisite, required, sufficient or optional.

swarm.security.security-domains.*KEY*.jaspi-authentication.auth-modules.*KEY*.login-module-stack-ref

Reference to a login module stack name previously configured in the same security domain.

swarm.security.security-domains.*KEY*.jaspi-authentication.auth-modules.*KEY*.module

Name of JBoss Module where the mapping module code is located.

swarm.security.security-domains.*KEY*.jaspi-authentication.auth-modules.*KEY*.module-options

List of module options containing a name/value pair.

swarm.security.security-domains.*KEY*.jaspi-authentication.login-module-stacks.*KEY*.login-modules.*KEY*.code

Class name of the module to be instantiated.

swarm.security.security-domains.*KEY*.jaspi-authentication.login-module-stacks.*KEY*.login-modules.*KEY*.flag

The flag controls how the module participates in the overall procedure. Allowed values are requisite, required, sufficient or optional.

swarm.security.security-domains.*KEY*.jaspi-authentication.login-module-stacks.*KEY*.login-modules.*KEY*.module

Name of JBoss Module where the login module is located.

swarm.security.security-domains.*KEY*.jaspi-authentication.login-module-stacks.*KEY*.login-modules.*KEY*.module-options

List of module options containing a name/value pair.

D.34. TOPOLOGY

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>topology</artifactId>
</dependency>
```

D.34.1. OpenShift

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>topology-openshift</artifactId>
</dependency>
```

D.34.2. Topology UI

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>topology-webapp</artifactId>
</dependency>
```

Configuration

swarm.topology.web-app.expose-topology-endpoint

Flag to enable or disable the topology web endpoint

swarm.topology.web-app.proxied-service-mappings

Service name to URL path proxy mappings

D.35. TRANSACTIONS

Provides support for the Java Transaction API (JTA) according to JSR-907.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>transactions</artifactId>
</dependency>
```

Configuration

swarm.transactions.average-commit-time

The average time of transaction commit in nanoseconds, measured from the moment the client calls commit until the transaction manager determines that the commit attempt was successful.

swarm.transactions.commit-markable-resources.*KEY*.batch-size

Batch size for this CMR resource

swarm.transactions.commit-markable-resources.*KEY*.immediate-cleanup

Immediate cleanup associated to this CMR resource

swarm.transactions.commit-markable-resources.*KEY*.jndi-name

JNDi name of this CMR resource

swarm.transactions.commit-markable-resources.*KEY*.name

table name for storing XIDs

swarm.transactions.default-timeout

The default timeout for a transaction managed by the transaction manager.

swarm.transactions.enable-statistics

Whether transaction statistics should be gathered.

swarm.transactions.enable-tsm-status

Whether the transaction status manager (TSM) service, needed for out of process recovery, should be provided or not.

swarm.transactions.hornetq-store-enable-async-io

Whether AsyncIO should be enabled for the journal store.

swarm.transactions.jdbc-action-store-drop-table

Configure if jdbc action store should drop tables.

swarm.transactions.jdbc-action-store-table-prefix

Optional prefix for table used to write transaction logs in configured jdbc action store.

swarm.transactions.jdbc-communication-store-drop-table

Configure if jdbc communication store should drop tables.

swarm.transactions.jdbc-communication-store-table-prefix

Optional prefix for table used to write transaction logs in configured jdbc communication store.

swarm.transactions.jdbc-state-store-drop-table

Configure if jdbc state store should drop tables.

swarm.transactions.jdbc-state-store-table-prefix

Optional prefix for table used to write transaction logs in configured jdbc state store.

swarm.transactions.jdbc-store-datasource

Jndi name of non-XA datasource used. Datasource sghould be define in datasources subsystem. For this would work the non-XA datasource has to be marked as jta="false".

swarm.transactions.journal-store-enable-async-io

Whether AsyncIO should be enabled for the journal store. For this settings being active journal natives libraries needs to be available.

swarm.transactions.jts

If true this enables the Java Transaction Service. Use of the JTS needs configuration in IIOP OpenJDK where Transactions parameter needs to be set to full.

swarm.transactions.log-store.expose-all-logs

Whether to expose all logs like orphans etc. By default only a subset of transaction logs is exposed.

swarm.transactions.log-store.transactions.*KEY*.age-in-seconds

The time since this transaction was prepared or when the recovery system last tried to recover it.

swarm.transactions.log-store.transactions.*KEY*.id

The id of this transaction.

swarm.transactions.log-store.transactions.*KEY*.jmx-name

The JMX name of this transaction.

swarm.transactions.log-store.transactions.*KEY*.participants.*KEY*.eis-product-name

The JCA enterprise information system's product name.

swarm.transactions.log-store.transactions.*KEY*.participants.*KEY*.eis-product-version

The JCA enterprise information system's product version

swarm.transactions.log-store.transactions.*KEY*.participants.*KEY*.jmx-name

The JMX name of this participant.

swarm.transactions.log-store.transactions.*KEY*.participants.*KEY*.jndi-name

JNDI name of this participant.

swarm.transactions.log-store.transactions.*KEY*.participants.*KEY*.status

Reports the commitment status of this participant (can be one of Pending, Prepared, Failed, Heuristic or Readonly).

swarm.transactions.log-store.transactions.*KEY*.participants.*KEY*.type

The type name under which this record is stored.

swarm.transactions.log-store.transactions.*KEY*.type

The type name under which this record is stored.

swarm.transactions.log-store.type

Specifies the implementation type of the logging store.

swarm.transactions.node-identifier

Used to set the node identifier on the core environment. Each Xid that Transaction Manager creates will have this identifier encoded within it and ensures Transaction Manager will only recover branches which match the specified identifier. It is imperative that this identifier is unique between Application Server instances which share either an object store or access common resource managers.

swarm.transactions.number-of-aborted-transactions

The number of aborted (i.e. rolledback) transactions.

swarm.transactions.number-of-application-rollbacks

The number of transactions that have been rolled back by application request. This includes those that timeout, since the timeout behavior is considered an attribute of the application configuration.

swarm.transactions.number-of-committed-transactions

The number of committed transactions.

swarm.transactions.number-of-heuristics

The number of transactions which have terminated with heuristic outcomes.

swarm.transactions.number-of-inflight-transactions

The number of transactions that have begun but not yet terminated.

swarm.transactions.number-of-nested-transactions

The total number of nested (sub) transactions created.

swarm.transactions.number-of-resource-rollbacks

The number of transactions that rolled back due to resource (participant) failure.

swarm.transactions.number-of-system-rollbacks

The number of transactions that have been rolled back due to internal system errors.

swarm.transactions.number-of-timed-out-transactions

The number of transactions that have rolled back due to timeout.

swarm.transactions.number-of-transactions

The total number of transactions (top-level and nested) created

swarm.transactions.object-store-path

Denotes a relative or absolute filesystem path denoting where the transaction manager object store should store data. By default the value is treated as relative to the path denoted by the "relative-to" attribute. This settings is valid when default or journal store is used. It’s not used when jdbc journal store is used.

swarm.transactions.object-store-relative-to

References a global path configuration in the domain model, defaulting to the Application Server data directory (jboss.server.data.dir). The value of the "Object store path" attribute will treated as relative to this path. Undefine this attribute to disable the default behavior and force the value of the "Object store path" attribute to be treated as an absolute path.

swarm.transactions.port

Port for transaction manager

swarm.transactions.process-id-socket-binding

The name of the socket binding configuration to use if the transaction manager should use a socket-based process id. Will be 'undefined' if 'process-id-uuid' is 'true'; otherwise must be set.

swarm.transactions.process-id-socket-max-ports

The maximum number of ports to search for an open port if the transaction manager should use a socket-based process id. If the port specified by the socket binding referenced in 'process-id-socket-binding' is occupied, the next higher port will be tried until an open port is found or the number of ports specified by this attribute have been tried. Will be 'undefined' if 'process-id-uuid' is 'true'.

swarm.transactions.process-id-uuid

Indicates whether the transaction manager should use a UUID based process id.

swarm.transactions.recovery-listener

Used to specify if the recovery system should listen on a network socket or not.

swarm.transactions.socket-binding

Used to reference the correct socket binding to use for the recovery environment.

swarm.transactions.statistics-enabled

Whether transaction statistics should be gathered.

swarm.transactions.status-port

Status port for transaction manager

swarm.transactions.status-socket-binding

Used to reference the correct socket binding to use for the transaction status manager.

swarm.transactions.use-hornetq-store

Use the journal store for writing transaction logs. Set to true to enable and to false to use the default log store type. The default log store is normally one file system file per transaction log.It’s alternative to jdbc based store.

swarm.transactions.use-jdbc-store

Use the jdbc store for writing transaction logs. Set to true to enable and to false to use the default log store type. The default log store is normally one file file per transaction log. It’s alternative to journal based store.

swarm.transactions.use-journal-store

Use the journal store for writing transaction logs. Set to true to enable and to false to use the default log store type. The default log store creates normally one file system file per transaction log. The journal one consists from one file for all the transactions. It’s alternative to jdbc based store.

D.36. UNDERTOW

Provides basic HTTP support, including Java Servlets, JavaServer Pages (JSP), and JavaServer Pages Standard Tag Library (JSTL) according to JSR-340, JSR-245 and JSR-52.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>undertow</artifactId>
</dependency>
```

Configuration

swarm.ajp.enable

Determine if AJP should be enabled

swarm.ajp.port

Set the port for the default AJP listener

swarm.deployment

Map of security configuration by deployment

swarm.http.port

Set the port for the default HTTP listener

swarm.https.certificate.generate

Should a self-signed certificate be generated

swarm.https.certificate.generate.host

Hostname for the generated self-signed certificate

swarm.https.key.alias

Alias to the server certificate key entry in the keystore

swarm.https.key.password

Password to the server certificate

swarm.https.keystore.embedded

Should an embedded keystore be created

swarm.https.keystore.password

Password to the server keystore

swarm.https.keystore.path

Path to the server keystore

swarm.https.only

Only enable the HTTPS Listener

swarm.https.port

Set the port for the default HTTPS listener

swarm.undertow.application-security-domains.*KEY*.enable-jacc

Enable authorization using JACC

swarm.undertow.application-security-domains.*KEY*.http-authentication-factory

The HTTP Authentication Factory to be used by deployments that reference the mapped security domain.

swarm.undertow.application-security-domains.*KEY*.override-deployment-config

Should the authentication configuration in the deployment be overridden by the factory.

swarm.undertow.application-security-domains.*KEY*.referencing-deployments

The deployments currently referencing this mapping.

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.client-ssl-context

Reference to the SSL context used to secure back-channel logout connection.

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.cookie-name

Name of the cookie

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.credential-reference

The credential reference to decrypt the private key entry.

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.domain

The cookie domain that will be used.

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.http-only

Set Cookie httpOnly attribute.

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.key-alias

Alias of the private key entry used for signing and verifying back-channel logout connection.

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.key-store

Reference to key store containing a private key entry.

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.path

Cookie path.

swarm.undertow.application-security-domains.*KEY*.single-sign-on-setting.secure

Set Cookie secure attribute.

swarm.undertow.buffer-caches.*KEY*.buffer-size

The size of an individual buffer, in bytes.

swarm.undertow.buffer-caches.*KEY*.buffers-per-region

The numbers of buffers in a region

swarm.undertow.buffer-caches.*KEY*.max-regions

The maximum number of regions

swarm.undertow.default-security-domain

The default security domain used by web deployments

swarm.undertow.default-server

The default server to use for deployments

swarm.undertow.default-servlet-container

The default servlet container to use for deployments

swarm.undertow.default-virtual-host

The default virtual host to use for deployments

swarm.undertow.filter-configuration.custom-filters.*KEY*.class-name

Class name of HttpHandler

swarm.undertow.filter-configuration.custom-filters.*KEY*.module

Module name where class can be loaded from

swarm.undertow.filter-configuration.custom-filters.*KEY*.parameters

Filter parameters

swarm.undertow.filter-configuration.error-pages.*KEY*.code

Error page code

swarm.undertow.filter-configuration.error-pages.*KEY*.path

Error page path

swarm.undertow.filter-configuration.expression-filters.*KEY*.expression

The expression that defines the filter

swarm.undertow.filter-configuration.expression-filters.*KEY*.module

Module to use to load the filter definitions

swarm.undertow.filter-configuration.mod-clusters.*KEY*.advertise-frequency

The frequency (in milliseconds) that mod-cluster advertises itself on the network

swarm.undertow.filter-configuration.mod-clusters.*KEY*.advertise-path

The path that mod-cluster is registered under.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.advertise-protocol

The protocol that is in use.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.advertise-socket-binding

The multicast group and port that is used to advertise.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.max-attempts

The number of attempts to send the request to a backend server

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.aliases

The nodes aliases

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.cache-connections

The number of connections to keep alive indefinitely

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.contexts.*KEY*.requests

The number of requests against this context

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.contexts.*KEY*.status

The status of this context

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.elected

The elected count

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.flush-packets

If received data should be immediately flushed

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.load

The current load of this node

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.load-balancing-group

The load balancing group this node belongs to

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.max-connections

The maximum number of connections per IO thread

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.open-connections

The current number of open connections

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.ping

The nodes ping

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.queue-new-requests

If a request is received and there is no worker immediately available should it be queued

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.read

The number of bytes read from the node

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.request-queue-size

The size of the request queue

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.status

The current status of this node

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.timeout

The request timeout

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.ttl

The time connections will stay alive with no requests before being closed, if the number of connections is larger than cache-connections

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.uri

The URI that the load balancer uses to connect to the node

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.written

The number of bytes transferred to the node

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.sticky-session

If sticky sessions are enabled

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.sticky-session-cookie

The session cookie name

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.sticky-session-force

If this is true then an error will be returned if the request cannot be routed to the sticky node, otherwise it will be routed to another node

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.sticky-session-path

The path of the sticky session cookie

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.sticky-session-remove

Remove the session cookie if the request cannot be routed to the correct host

swarm.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.wait-worker

The number of seconds to wait for an available worker

swarm.undertow.filter-configuration.mod-clusters.*KEY*.broken-node-timeout

The amount of time that must elapse before a broken node is removed from the table

swarm.undertow.filter-configuration.mod-clusters.*KEY*.cached-connections-per-thread

The number of connections that will be kept alive indefinitely

swarm.undertow.filter-configuration.mod-clusters.*KEY*.connection-idle-timeout

The amount of time a connection can be idle before it will be closed. Connections will not time out once the pool size is down to the configured minimum (as configured by cached-connections-per-thread)

swarm.undertow.filter-configuration.mod-clusters.*KEY*.connections-per-thread

The number of connections that will be maintained to backend servers, per IO thread.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.enable-http2

If the load balancer should attempt to upgrade back end connections to HTTP2. If HTTP2 is not supported HTTP or HTTPS will be used as normal

swarm.undertow.filter-configuration.mod-clusters.*KEY*.failover-strategy

Determines how a failover node is chosen, in the event that the node to which a session has affinity is not available.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.health-check-interval

The frequency of health check pings to backend nodes

swarm.undertow.filter-configuration.mod-clusters.*KEY*.http2-enable-push

If push should be enabled for HTTP/2 connections

swarm.undertow.filter-configuration.mod-clusters.*KEY*.http2-header-table-size

The size of the header table used for HPACK compression, in bytes. This amount of memory will be allocated per connection for compression. Larger values use more memory but may give better compression.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.http2-initial-window-size

The flow control window size that controls how quickly the client can send data to the server

swarm.undertow.filter-configuration.mod-clusters.*KEY*.http2-max-concurrent-streams

The maximum number of HTTP/2 streams that can be active at any time on a single connection

swarm.undertow.filter-configuration.mod-clusters.*KEY*.http2-max-frame-size

The max HTTP/2 frame size

swarm.undertow.filter-configuration.mod-clusters.*KEY*.http2-max-header-list-size

The maximum size of request headers the server is prepared to accept

swarm.undertow.filter-configuration.mod-clusters.*KEY*.management-access-predicate

A predicate that is applied to incoming requests to determine if they can perform mod cluster management commands. Provides additional security on top of what is provided by limiting management to requests that originate from the management-socket-binding

swarm.undertow.filter-configuration.mod-clusters.*KEY*.management-socket-binding

The socket binding of the mod_cluster management address and port. When using mod_cluster two HTTP listeners should be defined, a public one to handle requests, and one bound to the internal network to handle mod cluster commands. This socket binding should correspond to the internal listener, and should not be publicly accessible.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.max-ajp-packet-size

The maximum size for AJP packets. Increasing this will allow AJP to work for requests/responses that have a large amount of headers. This is an advanced option, and must be the same between load balancers and backend servers.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.max-request-time

The max amount of time that a request to a backend node can take before it is killed

swarm.undertow.filter-configuration.mod-clusters.*KEY*.max-retries

The number of times to attempt to retry a request if it fails. Note that if a request is not considered idempotent then it will only be retried if the proxy can be sure it was not sent to the backend server).

swarm.undertow.filter-configuration.mod-clusters.*KEY*.request-queue-size

The number of requests that can be queued if the connection pool is full before requests are rejected with a 503

swarm.undertow.filter-configuration.mod-clusters.*KEY*.security-key

The security key that is used for the mod-cluster group. All members must use the same security key.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.security-realm

The security realm that provides the SSL configuration

swarm.undertow.filter-configuration.mod-clusters.*KEY*.ssl-context

Reference to the SSLContext to be used by this filter.

swarm.undertow.filter-configuration.mod-clusters.*KEY*.use-alias

If an alias check is performed

swarm.undertow.filter-configuration.mod-clusters.*KEY*.worker

The XNIO worker that is used to send the advertise notifications

swarm.undertow.filter-configuration.request-limits.*KEY*.max-concurrent-requests

Maximum number of concurrent requests

swarm.undertow.filter-configuration.request-limits.*KEY*.queue-size

Number of requests to queue before they start being rejected

swarm.undertow.filter-configuration.response-headers.*KEY*.header-name

Header name

swarm.undertow.filter-configuration.response-headers.*KEY*.header-value

Value for header

swarm.undertow.filter-configuration.rewrites.*KEY*.redirect

If this is true then a redirect will be done instead of a rewrite

swarm.undertow.filter-configuration.rewrites.*KEY*.target

The expression that defines the target. If you are redirecting to a constant target put single quotes around the value

swarm.undertow.handler-configuration.files.*KEY*.cache-buffer-size

Size of the buffers, in bytes.

swarm.undertow.handler-configuration.files.*KEY*.cache-buffers

Number of buffers

swarm.undertow.handler-configuration.files.*KEY*.case-sensitive

Use case sensitive file handling

swarm.undertow.handler-configuration.files.*KEY*.directory-listing

Enable directory listing?

swarm.undertow.handler-configuration.files.*KEY*.follow-symlink

Enable following symbolic links

swarm.undertow.handler-configuration.files.*KEY*.path

Path on filesystem from where file handler will serve resources

swarm.undertow.handler-configuration.files.*KEY*.safe-symlink-paths

Paths that are safe to be targets of symbolic links

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.cached-connections-per-thread

The number of connections that will be kept alive indefinitely

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.connection-idle-timeout

The amount of time a connection can be idle before it will be closed. Connections will not time out once the pool size is down to the configured minimum (as configured by cached-connections-per-thread)

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.connections-per-thread

The number of connections that will be maintained to backend servers, per IO thread.

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.enable-http2

If this is true then the proxy will attempt to use HTTP/2 to connect to the backend. If it is not supported it will fall back to HTTP/1.1/

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.instance-id

The instance id (aka JVM route) that will be used to enable sticky sessions

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.outbound-socket-binding

Outbound socket binding for this host

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.path

Optional path if host is using non root resource

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.scheme

What kind of scheme is used

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.security-realm

The security realm that provides the SSL configuration for the connection to the host

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.ssl-context

Reference to the SSLContext to be used by this handler.

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.max-request-time

The maximum time that a proxy request can be active for, before being killed

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.max-retries

The number of times to attempt to retry a request if it fails. Note that if a request is not considered idempotent then it will only be retried if the proxy can be sure it was not sent to the backend server).

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.problem-server-retry

Time in seconds to wait before attempting to reconnect to a server that is down

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.request-queue-size

The number of requests that can be queued if the connection pool is full before requests are rejected with a 503

swarm.undertow.handler-configuration.reverse-proxies.*KEY*.session-cookie-names

Comma separated list of session cookie names. Generally this will just be JSESSIONID.

swarm.undertow.instance-id

The cluster instance id

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.allow-encoded-slash

If a request comes in with encoded / characters (i.e. %2F), will these be decoded.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.allow-equals-in-cookie-value

If this is true then Undertow will allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.always-set-keep-alive

If this is true then a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.buffer-pipelined-data

If we should buffer pipelined requests.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.buffer-pool

The listeners buffer pool

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.bytes-received

The number of bytes that have been received by this listener

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.bytes-sent

The number of bytes that have been sent out on this listener

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.decode-url

If this is true then the parser will decode the URL and query parameters using the selected character encoding (UTF-8 by default). If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.disallowed-methods

A comma separated list of HTTP methods that are not allowed

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.error-count

The number of 500 responses that have been sent by this listener

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-ajp-packet-size

The maximum supported size of AJP packets. If this is modified it has to be increased on the load balancer and the backend server.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-buffered-request-size

Maximum size of a buffered request, in bytes. Requests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-connections

The maximum number of concurrent connections. Only values greater than 0 are allowed. For unlimited connections simply undefine this attribute value.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-cookies

The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-header-size

The maximum size of a http request header, in bytes.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-headers

The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-parameters

The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative (i.e. you can potentially have max parameters * 2 total parameters).

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-post-size

The maximum size of a post that will be accepted, in bytes.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-processing-time

The maximum processing time taken by a request on this listener

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.no-request-timeout

The length of time in milliseconds that the connection can be idle before it is closed by the container.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.processing-time

The total processing time of all requests handed by this listener

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.read-timeout

Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a {@link ReadTimeoutException}.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.receive-buffer

The receive buffer size, in bytes.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.record-request-start-time

If this is true then Undertow will record the request start time, to allow for request time to be logged. This has a small but measurable performance impact

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.redirect-socket

If this listener is supporting non-SSL requests, and a request is received for which a matching <security-constraint> requires SSL transport, undertow will automatically redirect the request to the socket binding port specified here.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.request-count

The number of requests this listener has served

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.request-parse-timeout

The maximum amount of time (in milliseconds) that can be spent parsing the request

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.resolve-peer-address

Enables host dns lookup

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.rfc6265-cookie-validation

If cookies should be validated to ensure they comply with RFC6265.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.scheme

The listener scheme, can be HTTP or HTTPS. By default the scheme will be taken from the incoming AJP request.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.secure

If this is true then requests that originate from this listener are marked as secure, even if the request is not using HTTPS.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.send-buffer

The send buffer size, in bytes.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.socket-binding

The listener socket binding

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.tcp-backlog

Configure a server with the specified backlog.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.tcp-keep-alive

Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.url-charset

URL charset

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.worker

The listeners XNIO worker

swarm.undertow.servers.*KEY*.ajp-listeners.*KEY*.write-timeout

Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a {@link WriteTimeoutException}.

swarm.undertow.servers.*KEY*.default-host

The servers default virtual host

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.directory

Directory in which to save logs

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.extended

If the log uses the extended log file format

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.pattern

The access log pattern.

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.predicate

Predicate that determines if the request should be logged

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.prefix

Prefix for the log file name.

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.relative-to

The directory the path is relative to

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.rotate

Rotate the access log every day.

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.suffix

Suffix for the log file name.

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.use-server-log

If the log should be written to the server log, rather than a separate file.

swarm.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.worker

Name of the worker to use for logging

swarm.undertow.servers.*KEY*.hosts.*KEY*.alias

Aliases for the host

swarm.undertow.servers.*KEY*.hosts.*KEY*.default-response-code

If set, this will be response code sent back in case requested context does not exist on server.

swarm.undertow.servers.*KEY*.hosts.*KEY*.default-web-module

Default web module

swarm.undertow.servers.*KEY*.hosts.*KEY*.disable-console-redirect

if set to true, /console redirect wont be enabled for this host, default is false

swarm.undertow.servers.*KEY*.hosts.*KEY*.filter-refs.*KEY*.predicate

Predicates provide a simple way of making a true/false decision based on an exchange. Many handlers have a requirement that they be applied conditionally, and predicates provide a general way to specify a condition.

swarm.undertow.servers.*KEY*.hosts.*KEY*.filter-refs.*KEY*.priority

Defines filter order, it should be set to 1 or more, higher number instructs server to be included earlier in handler chain than others under same context.

swarm.undertow.servers.*KEY*.hosts.*KEY*.http-invoker-setting.http-authentication-factory

The HTTP authentication factory to use for authentication

swarm.undertow.servers.*KEY*.hosts.*KEY*.http-invoker-setting.path

The path that the services are installed under

swarm.undertow.servers.*KEY*.hosts.*KEY*.http-invoker-setting.security-realm

The legacy security realm to use for authentication

swarm.undertow.servers.*KEY*.hosts.*KEY*.locations.*KEY*.filter-refs.*KEY*.predicate

Predicates provide a simple way of making a true/false decision based on an exchange. Many handlers have a requirement that they be applied conditionally, and predicates provide a general way to specify a condition.

swarm.undertow.servers.*KEY*.hosts.*KEY*.locations.*KEY*.filter-refs.*KEY*.priority

Defines filter order, it should be set to 1 or more, higher number instructs server to be included earlier in handler chain than others under same context.

swarm.undertow.servers.*KEY*.hosts.*KEY*.locations.*KEY*.handler

Default handler for this location

swarm.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.cookie-name

Name of the cookie

swarm.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.domain

The cookie domain that will be used.

swarm.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.http-only

Set Cookie httpOnly attribute.

swarm.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.path

Cookie path.

swarm.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.secure

Set Cookie secure attribute.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.allow-encoded-slash

If a request comes in with encoded / characters (i.e. %2F), will these be decoded.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.allow-equals-in-cookie-value

If this is true then Undertow will allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.always-set-keep-alive

If this is true then a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.buffer-pipelined-data

If we should buffer pipelined requests.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.buffer-pool

The listeners buffer pool

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.bytes-received

The number of bytes that have been received by this listener

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.bytes-sent

The number of bytes that have been sent out on this listener

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.certificate-forwarding

If certificate forwarding should be enabled. If this is enabled then the listener will take the certificate from the SSL_CLIENT_CERT attribute. This should only be enabled if behind a proxy, and the proxy is configured to always set these headers.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.decode-url

If this is true then the parser will decode the URL and query parameters using the selected character encoding (UTF-8 by default). If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.disallowed-methods

A comma separated list of HTTP methods that are not allowed

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.enable-http2

Enables HTTP2 support for this listener

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.error-count

The number of 500 responses that have been sent by this listener

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.http2-enable-push

If server push is enabled for this connection

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.http2-header-table-size

The size of the header table used for HPACK compression, in bytes. This amount of memory will be allocated per connection for compression. Larger values use more memory but may give better compression.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.http2-initial-window-size

The flow control window size that controls how quickly the client can send data to the server

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.http2-max-concurrent-streams

The maximum number of HTTP/2 streams that can be active at any time on a single connection

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.http2-max-frame-size

The max HTTP/2 frame size

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.http2-max-header-list-size

The maximum size of request headers the server is prepared to accept

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.max-buffered-request-size

Maximum size of a buffered request, in bytes. Requests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.max-connections

The maximum number of concurrent connections. Only values greater than 0 are allowed. For unlimited connections simply undefine this attribute value.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.max-cookies

The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.max-header-size

The maximum size of a http request header, in bytes.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.max-headers

The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.max-parameters

The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative (i.e. you can potentially have max parameters * 2 total parameters).

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.max-post-size

The maximum size of a post that will be accepted, in bytes.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.max-processing-time

The maximum processing time taken by a request on this listener

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.no-request-timeout

The length of time in milliseconds that the connection can be idle before it is closed by the container.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.processing-time

The total processing time of all requests handed by this listener

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.proxy-address-forwarding

Enables handling of x-forwarded-host header (and other x-forwarded-* headers) and use this header information to set the remote address. This should only be used behind a trusted proxy that sets these headers otherwise a remote user can spoof their IP address.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.read-timeout

Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a {@link ReadTimeoutException}.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.receive-buffer

The receive buffer size, in bytes.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.record-request-start-time

If this is true then Undertow will record the request start time, to allow for request time to be logged. This has a small but measurable performance impact

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.redirect-socket

If this listener is supporting non-SSL requests, and a request is received for which a matching <security-constraint> requires SSL transport, undertow will automatically redirect the request to the socket binding port specified here.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.request-count

The number of requests this listener has served

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.request-parse-timeout

The maximum amount of time (in milliseconds) that can be spent parsing the request

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.require-host-http11

Require that all HTTP/1.1 requests have a 'Host' header, as per the RFC. IF the request does not include this header it will be rejected with a 403.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.resolve-peer-address

Enables host dns lookup

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.rfc6265-cookie-validation

If cookies should be validated to ensure they comply with RFC6265.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.secure

If this is true then requests that originate from this listener are marked as secure, even if the request is not using HTTPS.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.send-buffer

The send buffer size, in bytes.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.socket-binding

The listener socket binding

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.tcp-backlog

Configure a server with the specified backlog.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.tcp-keep-alive

Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.url-charset

URL charset

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.worker

The listeners XNIO worker

swarm.undertow.servers.*KEY*.http-listeners.*KEY*.write-timeout

Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a {@link WriteTimeoutException}.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.allow-encoded-slash

If a request comes in with encoded / characters (i.e. %2F), will these be decoded.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.allow-equals-in-cookie-value

If this is true then Undertow will allow non-escaped equals characters in unquoted cookie values. Unquoted cookie values may not contain equals characters. If present the value ends before the equals sign. The remainder of the cookie value will be dropped.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.always-set-keep-alive

If this is true then a Connection: keep-alive header will be added to responses, even when it is not strictly required by the specification.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.buffer-pipelined-data

If we should buffer pipelined requests.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.buffer-pool

The listeners buffer pool

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.bytes-received

The number of bytes that have been received by this listener

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.bytes-sent

The number of bytes that have been sent out on this listener

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.certificate-forwarding

If certificate forwarding should be enabled. If this is enabled then the listener will take the certificate from the SSL_CLIENT_CERT attribute. This should only be enabled if behind a proxy, and the proxy is configured to always set these headers.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.decode-url

If this is true then the parser will decode the URL and query parameters using the selected character encoding (UTF-8 by default). If this is false they will not be decoded. This will allow a later handler to decode them into whatever charset is desired.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.disallowed-methods

A comma separated list of HTTP methods that are not allowed

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.enable-http2

Enables HTTP2 support for this listener

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.enabled-cipher-suites

Configures Enabled SSL ciphers

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.enabled-protocols

Configures SSL protocols

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.error-count

The number of 500 responses that have been sent by this listener

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.http2-enable-push

If server push is enabled for this connection

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.http2-header-table-size

The size of the header table used for HPACK compression, in bytes. This amount of memory will be allocated per connection for compression. Larger values use more memory but may give better compression.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.http2-initial-window-size

The flow control window size that controls how quickly the client can send data to the server

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.http2-max-concurrent-streams

The maximum number of HTTP/2 streams that can be active at any time on a single connection

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.http2-max-frame-size

The max HTTP/2 frame size

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.http2-max-header-list-size

The maximum size of request headers the server is prepared to accept

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.max-buffered-request-size

Maximum size of a buffered request, in bytes. Requests are not usually buffered, the most common case is when performing SSL renegotiation for a POST request, and the post data must be fully buffered in order to perform the renegotiation.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.max-connections

The maximum number of concurrent connections. Only values greater than 0 are allowed. For unlimited connections simply undefine this attribute value.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.max-cookies

The maximum number of cookies that will be parsed. This is used to protect against hash vulnerabilities.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.max-header-size

The maximum size of a http request header, in bytes.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.max-headers

The maximum number of headers that will be parsed. This is used to protect against hash vulnerabilities.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.max-parameters

The maximum number of parameters that will be parsed. This is used to protect against hash vulnerabilities. This applies to both query parameters, and to POST data, but is not cumulative (i.e. you can potentially have max parameters * 2 total parameters).

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.max-post-size

The maximum size of a post that will be accepted, in bytes.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.max-processing-time

The maximum processing time taken by a request on this listener

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.no-request-timeout

The length of time in milliseconds that the connection can be idle before it is closed by the container.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.processing-time

The total processing time of all requests handed by this listener

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.proxy-address-forwarding

Enables handling of x-forwarded-host header (and other x-forwarded-* headers) and use this header information to set the remote address. This should only be used behind a trusted proxy that sets these headers otherwise a remote user can spoof their IP address.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.read-timeout

Configure a read timeout for a socket, in milliseconds. If the given amount of time elapses without a successful read taking place, the socket's next read will throw a {@link ReadTimeoutException}.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.receive-buffer

The receive buffer size, in bytes.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.record-request-start-time

If this is true then Undertow will record the request start time, to allow for request time to be logged. This has a small but measurable performance impact

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.request-count

The number of requests this listener has served

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.request-parse-timeout

The maximum amount of time (in milliseconds) that can be spent parsing the request

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.require-host-http11

Require that all HTTP/1.1 requests have a 'Host' header, as per the RFC. IF the request does not include this header it will be rejected with a 403.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.resolve-peer-address

Enables host dns lookup

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.rfc6265-cookie-validation

If cookies should be validated to ensure they comply with RFC6265.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.secure

If this is true then requests that originate from this listener are marked as secure, even if the request is not using HTTPS.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.security-realm

The listeners security realm

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.send-buffer

The send buffer size, in bytes.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.socket-binding

The listener socket binding

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.ssl-context

Reference to the SSLContext to be used by this listener.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.ssl-session-cache-size

The maximum number of active SSL sessions

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.ssl-session-timeout

The timeout for SSL sessions, in seconds

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.tcp-backlog

Configure a server with the specified backlog.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.tcp-keep-alive

Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.url-charset

URL charset

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.verify-client

The desired SSL client authentication mode for SSL channels

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.worker

The listeners XNIO worker

swarm.undertow.servers.*KEY*.https-listeners.*KEY*.write-timeout

Configure a write timeout for a socket, in milliseconds. If the given amount of time elapses without a successful write taking place, the socket's next write will throw a {@link WriteTimeoutException}.

swarm.undertow.servers.*KEY*.servlet-container

The servers default servlet container

swarm.undertow.servlet-containers.*KEY*.allow-non-standard-wrappers

If true then request and response wrappers that do not extend the standard wrapper classes can be used

swarm.undertow.servlet-containers.*KEY*.crawler-session-management-setting.session-timeout

The session timeout for sessions that are owned by crawlers

swarm.undertow.servlet-containers.*KEY*.crawler-session-management-setting.user-agents

Regular expression that is used to match the user agent of a crawler

swarm.undertow.servlet-containers.*KEY*.default-buffer-cache

The buffer cache to use for caching static resources

swarm.undertow.servlet-containers.*KEY*.default-encoding

Default encoding to use for all deployed applications

swarm.undertow.servlet-containers.*KEY*.default-session-timeout

The default session timeout (in minutes) for all applications deployed in the container.

swarm.undertow.servlet-containers.*KEY*.directory-listing

If directory listing should be enabled for default servlets.

swarm.undertow.servlet-containers.*KEY*.disable-caching-for-secured-pages

If Undertow should set headers to disable caching for secured paged. Disabling this can cause security problems, as sensitive pages may be cached by an intermediary.

swarm.undertow.servlet-containers.*KEY*.disable-file-watch-service

If this is true then the file watch service will not be used to monitor exploded deployments for changes

swarm.undertow.servlet-containers.*KEY*.disable-session-id-reuse

If this is true then an unknown session ID will never be reused, and a new session id will be generated. If this is false then it will be re-used if and only if it is present in the session manager of another deployment, to allow the same session id to be shared between applications on the same server.

swarm.undertow.servlet-containers.*KEY*.eager-filter-initialization

If true undertow calls filter init() on deployment start rather than when first requested.

swarm.undertow.servlet-containers.*KEY*.ignore-flush

Ignore flushes on the servlet output stream. In most cases these just hurt performance for no good reason.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.check-interval

Check interval for JSP updates using a background thread. This has no effect for most deployments where JSP change notifications are handled using the File System notification API. This only takes effect if the file watch service is disabled.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.development

Enable Development mode which enables reloading JSP on-the-fly

swarm.undertow.servlet-containers.*KEY*.jsp-setting.disabled

Disable the JSP container.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.display-source-fragment

When a runtime error occurs, attempts to display corresponding JSP source fragment

swarm.undertow.servlet-containers.*KEY*.jsp-setting.dump-smap

Write SMAP data to a file.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.error-on-use-bean-invalid-class-attribute

Enable errors when using a bad class in useBean.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.generate-strings-as-char-arrays

Generate String constants as char arrays.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.java-encoding

Specify the encoding used for Java sources.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.keep-generated

Keep the generated Servlets.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.mapped-file

Map to the JSP source.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.modification-test-interval

Minimum amount of time between two tests for updates, in seconds.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.optimize-scriptlets

If JSP scriptlets should be optimised to remove string concatenation

swarm.undertow.servlet-containers.*KEY*.jsp-setting.recompile-on-fail

Retry failed JSP compilations on each request.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.scratch-dir

Specify a different work directory.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.smap

Enable SMAP.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.source-vm

Source VM level for compilation.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.tag-pooling

Enable tag pooling.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.target-vm

Target VM level for compilation.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.trim-spaces

Trim some spaces from the generated Servlet.

swarm.undertow.servlet-containers.*KEY*.jsp-setting.xPowered-by

Enable advertising the JSP engine in x-powered-by.

swarm.undertow.servlet-containers.*KEY*.max-sessions

The maximum number of sessions that can be active at one time

swarm.undertow.servlet-containers.*KEY*.mime-mappings.*KEY*.value

The mime type for this mapping

swarm.undertow.servlet-containers.*KEY*.persistent-sessions-setting.path

The path to the persistent session data directory. If this is null sessions will be stored in memory

swarm.undertow.servlet-containers.*KEY*.persistent-sessions-setting.relative-to

The directory the path is relative to

swarm.undertow.servlet-containers.*KEY*.proactive-authentication

If proactive authentication should be used. If this is true a user will always be authenticated if credentials are present.

swarm.undertow.servlet-containers.*KEY*.session-cookie-setting.comment

Cookie comment

swarm.undertow.servlet-containers.*KEY*.session-cookie-setting.domain

Cookie domain

swarm.undertow.servlet-containers.*KEY*.session-cookie-setting.http-only

Is cookie http-only

swarm.undertow.servlet-containers.*KEY*.session-cookie-setting.max-age

Max age of cookie

swarm.undertow.servlet-containers.*KEY*.session-cookie-setting.name

Name of the cookie

swarm.undertow.servlet-containers.*KEY*.session-cookie-setting.secure

Is cookie secure?

swarm.undertow.servlet-containers.*KEY*.session-id-length

The length of the generated session ID. Longer session ID's are more secure. This number refers to the number of bytes of randomness that are used to generate the session ID, the actual ID that is sent to the client will be base64 encoded so will be approximately 33% larger (e.g. a session id length of 30 will result in a cookie value of length 40).

swarm.undertow.servlet-containers.*KEY*.stack-trace-on-error

If an error page with the stack trace should be generated on error. Values are all, none and local-only

swarm.undertow.servlet-containers.*KEY*.use-listener-encoding

Use encoding defined on listener

swarm.undertow.servlet-containers.*KEY*.websockets-setting.buffer-pool

The buffer pool to use for websocket deployments

swarm.undertow.servlet-containers.*KEY*.websockets-setting.deflater-level

Configures the level of compression of the DEFLATE algorithm

swarm.undertow.servlet-containers.*KEY*.websockets-setting.dispatch-to-worker

If callbacks should be dispatched to a worker thread. If this is false then they will be run in the IO thread, which is faster however care must be taken not to perform blocking operations.

swarm.undertow.servlet-containers.*KEY*.websockets-setting.per-message-deflate

Enables websocket's per-message compression extension, RFC-7692

swarm.undertow.servlet-containers.*KEY*.websockets-setting.worker

The worker to use for websocket deployments

swarm.undertow.statistics-enabled

Configures if statistics are enabled. Changes take effect on the connector level statistics immediately, deployment level statistics will only be affected after the deployment is redeployed (or the container is reloaded).

D.37. WEB

Provides a collection of fractions equivalent to the *Web Profile*:

- Bean Validation
- CDI
- EJB
- JAX-RS

    - JSON-P
    - JAXB
    - Multipart
    - Validator
- JPA
- JSF
- Transactions
- Undertow (Servlets)

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>web</artifactId>
</dependency>
```

APPENDIX E. ADDITIONAL THORNTAIL RESOURCES

- Thorntail Community Documentation
- Thorntail Presentations
- Thorntail, Microservices & OpenShift Lab
- Thorntail Microservices On Red Hat OpenShift Container Platform 3

APPENDIX F. APPLICATION DEVELOPMENT RESOURCES

For additional information on application development with OpenShift see:

- OpenShift Interactive Learning Portal
- Red Hat OpenShift Application Runtimes Overview

APPENDIX G. PROFICIENCY LEVELS

Each available mission teaches concepts that require certain minimum knowledge. This requirement varies by mission. The minimum requirements and concepts are organized in several levels of proficiency. In addition to the levels described here, you might need additional information specific to each mission.

Foundational

The missions rated at Foundational proficiency generally require no prior knowledge of the subject matter; they provide general awareness and demonstration of key elements, concepts, and terminology. There are no special requirements except those directly mentioned in the description of the mission.

Advanced

When using Advanced missions, the assumption is that you are familiar with the common concepts and terminology of the subject area of the mission in addition to Kubernetes and OpenShift. You must also be able to perform basic tasks on your own, for example configure services and applications, or administer networks. If a service is needed by the mission, but configuring it is not in the scope of the mission, the assumption is that you have the knowledge to to properly configure it, and only the resulting state of the service is described in the documentation.

Expert

Expert missions require the highest level of knowledge of the subject matter. You are expected to perform many tasks based on feature-based documentation and manuals, and the documentation is aimed at most complex scenarios.

APPENDIX H. GLOSSARY

H.1. PRODUCT AND PROJECT NAMES

developers.redhat.com/launch

developers.redhat.com/launch is a standalone getting started experience offered by Red Hat for jumpstarting cloud-native application development on OpenShift. It provides a hassle-free way of creating functional example applications, called missions, as well as an easy way to build and deploy those missions to OpenShift.

Fabric8 Launcher

The Fabric8 Launcher is the upstream project on which developers.redhat.com/launch is based.

Single-node OpenShift Cluster

An OpenShift cluster running on your machine using Minishift.

H.2. TERMS SPECIFIC TO FABRIC8 LAUNCHER

Booster

A language-specific implementation of a particular mission on a particular runtime. Boosters are listed in a booster catalog.
For example, a booster is a web service with a REST API implemented using the Thorntail runtime.

Booster Catalog

A Git repository that contains information about boosters.

Mission

An application specification, for example *a web service with a REST API.*
Missions generally do not specify which language or platform they should run on; the description only contains the intended functionality.

Runtime

A platform that executes boosters. For example, Thorntail or Eclipse Vert.x.