Red Hat build of Thorntail 2.7

Thorntail Runtime Guide

Use Thorntail to develop small, stand-alone, microservice-based applications that run on OpenShift and on stand-alone RHEL

Last Updated: 2021-05-04

Red Hat build of Thorntail 2.7 Thorntail Runtime Guide

Use Thorntail to develop small, stand-alone, microservice-based applications that run on OpenShift and on stand-alone RHEL

Legal Notice

Abstract

This guide provides details about using Thorntail.

Table of Contents

PREFACE

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. To provide feedback, you can highlight the text in a document and add comments.

This section explains how to submit feedback.

Prerequisites

- You are logged in to the Red Hat Customer Portal.
- In the Red Hat Customer Portal, view the document in **Multi-page HTML** format.

Procedure

To provide your feedback, perform the following steps:

1. Click the **Feedback** button in the top-right corner of the document to see existing feedback.

   NOTE

   The feedback feature is enabled only in the **Multi-page HTML** format.

2. Highlight the section of the document where you want to provide feedback.
3. Click the **Add Feedback** pop-up that appears near the highlighted text.
   A text box appears in the feedback section on the right side of the page.
4. Enter your feedback in the text box and click **Submit**.
   A documentation issue is created.
5. To view the issue, click the issue tracker link in the feedback view.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message .

CHAPTER 1. INTRODUCTION TO APPLICATION DEVELOPMENT WITH THORNTAIL

This section explains the basic concepts of application development with Red Hat runtimes. It also provides an overview about the Thorntail runtime.

1.1. OVERVIEW OF APPLICATION DEVELOPMENT WITH RED HAT RUNTIMES

Red Hat OpenShift is a container application platform, which provides a collection of cloud-native runtimes. You can use the runtimes to develop, build, and deploy Java or JavaScript applications on OpenShift.

Application development using Red Hat Runtimes for OpenShift includes:

- A collection of runtimes, such as, Eclipse Vert.x, Thorntail, Spring Boot, and so on, designed to run on OpenShift.
- A prescriptive approach to cloud-native development on OpenShift.

OpenShift helps you manage, secure, and automate the deployment and monitoring of your applications. You can break your business problems into smaller microservices and use OpenShift to deploy, monitor, and maintain the microservices. You can implement patterns such as circuit breaker, health check, and service discovery, in your applications.

Cloud-native development takes full advantage of cloud computing.

You can build, deploy, and manage your applications on:

OpenShift Container Platform

A private on-premise cloud by Red Hat.

Red Hat Container Development Kit (Minishift)

A local cloud that you can install and execute on your local machine. This functionality is provided by Red Hat Container Development Kit (CDK) or Minishift.

Red Hat CodeReady Studio

An integrated development environment (IDE) for developing, testing, and deploying applications.

To help you get started with application development, all the runtimes are available with example applications. These example applications are accessible from the Developer Launcher. You can use the examples as templates to create your applications.

This guide provides detailed information about the Thorntail runtime. For more information on other runtimes, see the relevant runtime documentation.

1.2. APPLICATION DEVELOPMENT ON RED HAT OPENSHIFT USING DEVELOPER LAUNCHER

You can get started with developing cloud-native applications on OpenShift using Developer Launcher (developers.redhat.com/launch). It is a service provided by Red Hat.

Developer Launcher is a stand-alone project generator. You can use it to build and deploy applications on OpenShift instances, such as, OpenShift Container Platform or Minishift or CDK.

1.2.1. Supported Architectures by Thorntail

Thorntail supports the following architectures:

- x86_64 (AMD64)
- IBM Z (s390x) in the OpenShift environment
- IBM Power Systems (ppc64le) in the OpenShift environment

Different images are supported for different architectures. The example codes in this guide demonstrate the commands for x86_64 architecture. If you are using other architectures, specify the relevant image name in the commands.

Refer to the section Supported Java images for Thorntail for more information about the image names.

1.2.2. Introduction to example applications

Examples are working applications that demonstrate how to build cloud native applications and services. They demonstrate prescriptive architectures, design patterns, tools, and best practices that should be used when you develop your applications. The example applications can be used as templates to create your cloud-native microservices. You can update and redeploy these examples using the deployment process explained in this guide.

The examples implement Microservice patterns such as:

- Creating REST APIs
- Interoperating with a database
- Implementing the health check pattern
- Externalizing the configuration of your applications to make them more secure and easier to scale

You can use the examples applications as:

- Working demonstration of the technology
- Learning tool or a sandbox to understand how to develop applications for your project
- Starting point for updating or extending your own use case

Each example application is implemented in one or more runtimes. For example, the REST API Level 0 example is available for the following runtimes:

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

The subsequent sections explain the example applications implemented for the Thorntail runtime.

You can download and deploy all the example applications on:

- x86_64 architecture - The example applications in this guide demonstrate how to build and deploy example applications on x86_64 architecture.
- s390x architecture - To deploy the example applications on OpenShift environments provisioned on IBM Z infrastructure, specify the relevant IBM Z image name in the commands.
- ppc64le architecture - To deploy the example applications on OpenShift environments provisioned on IBM Power Systems infrastructure, specify the relevant IBM Power Systems image name in the commands.

Refer to the section Supported Java images for Thorntail for more information about the image names.

Some of the example applications also require other products, such as Red Hat Data Grid to demonstrate the workflows. In this case, you must also change the image names of these products to their relevant IBM Z and IBM Power Systems image names in the YAML file of the example applications.

CHAPTER 2. DOWNLOADING AND DEPLOYING APPLICATIONS USING DEVELOPER LAUNCHER

This section shows you how to download and deploy example applications provided with the runtimes. The example applications are available on Developer Launcher.

2.1. WORKING WITH DEVELOPER LAUNCHER

Developer Launcher (developers.redhat.com/launch) runs on OpenShift. When you deploy example applications, the Developer Launcher guides you through the process of:

- Selecting a runtime
- Building and executing the application

Based on your selection, Developer Launcher generates a custom project. You can either download a ZIP version of the project or directly launch the application on an OpenShift Online instance.

When you deploy your application on OpenShift using Developer Launcher, the Source-to-Image (S2I) build process is used. This build process handles all the configuration, build, and deployment steps that are required to run your application on OpenShift.

2.2. DOWNLOADING THE EXAMPLE APPLICATIONS USING DEVELOPER LAUNCHER

Red Hat provides example applications that help you get started with the Thorntail runtime. These examples are available on Developer Launcher (developers.redhat.com/launch).

You can download the example applications, build, and deploy them. This section explains how to download example applications.

You can use the example applications as templates to create your own cloud-native applications.

Procedure

1. Go to Developer Launcher (developers.redhat.com/launch).
2. Click *Start*.
3. Click *Deploy an Example Application*.
4. Click *Select an Example* to see the list of example applications available with the runtime.
5. Select a runtime.
6. Select an example application.

> **NOTE**
>
> Some example applications are available for multiple runtimes. If you have not selected a runtime in the previous step, you can select a runtime from the list of available runtimes in the example application.

7. Select the release version for the runtime. You can choose from the community or product releases listed for the runtime.

8. Click *Save*.

9. Click *Download* to download the example application.
   A ZIP file containing the source and documentation files is downloaded.

2.3. DEPLOYING AN EXAMPLE APPLICATION ON OPENSHIFT CONTAINER PLATFORM OR CDK (MINISHIFT)

You can deploy the example application to either OpenShift Container Platform or CDK (Minishift). Depending on where you want to deploy your application use the relevant web console for authentication.

Prerequisites

- An example application project created using Developer Launcher.

- If you are deploying your application on OpenShift Container Platform, you must have access to the OpenShift Container Platform web console.

- If you are deploying your application on CDK (Minishift), you must have access to the CDK (Minishift) web console.

- **oc** command-line client installed.

Procedure

1. Download the example application.

2. You can deploy the example application on OpenShift Container Platform or CDK (Minishift) using the **oc** command-line client.
   You must authenticate the client using the token provided by the web console. Depending on where you want to deploy your application, use either the OpenShift Container Platform web console or CDK (Minishift) web console. Perform the following steps to get the authenticate the client:

   a. Login to the web console.

   b. Click the question mark icon, which is in the upper-right corner of the web console.

   c. Select *Command Line Tools* from the list.

   d. Copy the **oc login** command.

   e. Paste the command in a terminal to authenticate your **oc** CLI client with your account.

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

3. Extract the contents of the ZIP file.

   ```
   $ unzip MY_APPLICATION_NAME.zip
   ```

4. Create a new project in OpenShift.

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

5. Navigate to the root directory of **MY_APPLICATION_NAME**.

6. Deploy your example application using Maven.

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

   NOTE: Some example applications may require additional setups. To build and deploy the example applications, follow the instructions provided in the **README** file.

7. Check the status of your application 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 has the status **Running** after it is fully deployed and started. The pod name of your application may be different. The numeric value in the pod name is incremented for every new build. The letters at the end are generated when the pod is created.

8. After your example application 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 can use to access it. In this example, you can use **http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME** as the base URL to access the application.

CHAPTER 3. DEVELOPING AND DEPLOYING THORNTAIL APPLICATION

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

3.1. CREATING AN APPLICATION FROM SCRATCH

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

Prerequisites

- OpenJDK 8 or OpenJDK 11 installed
- Maven 3.6.0 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>2.7.3.Final-redhat-00001</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>

    <!-- Specify the JDK builder image used to build your application. -->
    <jkube.generator.from>registry.access.redhat.com/redhat-openjdk-18/openjdk18-
openshift:latest</jkube.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>

</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
```

3.2. DEPLOYING THORNTAIL APPLICATION TO OPENSHIFT

To deploy your Thorntail application to OpenShift, configure the **pom.xml** file in your application and then use the OpenShift Maven plugin.

> **NOTE**
>
> The Fabric8 Maven plugin is no longer supported. Use the OpenShift Maven plugin to deploy your Thorntail applications on OpenShift. For more information, see the section migrating from Fabric8 Maven Plugin to Eclipse JKube .

You can specify a Java image by replacing the **jkube.generator.from** URL in the **pom.xml** file. The images are available in the Red Hat Ecosystem Catalog .

```
<jkube.generator.from>IMAGE_NAME</jkube.generator.from>
```

For example, the Java image for RHEL 7 with OpenJDK 8 is specified as:

```
<jkube.generator.from>registry.access.redhat.com/redhat-openjdk-18/openjdk18-
openshift:latest</jkube.generator.from>
```

3.2.1. Supported Java images for Thorntail

Thorntail is certified and tested with various Java images that are available for different operating systems. For example, Java images are available for RHEL 7 and RHEL 8 with OpenJDK 8 or OpenJDK 11. Similar images are available on IBM Z and IBM Power Systems.

You require Docker or podman authentication to access the RHEL 8 images in the Red Hat Ecosystem Catalog.

The following table lists the images supported by Thorntail for different architectures. It also provides links to the images available in the Red Hat Ecosystem Catalog. The image pages contain authentication procedures required to access the RHEL 8 images.

3.2.1.1. Images on x86_64 architecture

OS	Java	Red Hat Ecosystem Catalog
RHEL 7	OpenJDK 8	RHEL 7 with OpenJDK 8
RHEL 7	OpenJDK 11	RHEL 7 with OpenJDK 11
RHEL 8	OpenJDK 8	RHEL 8 with OpenJDK 8
RHEL 8	OpenJDK 11	RHEL 8 with OpenJDK 11

NOTE

The use of a RHEL 8-based container on a RHEL 7 host, for example with OpenShift 3 or OpenShift 4, has limited support. For more information, see the Red Hat Enterprise Linux Container Compatibility Matrix.

3.2.1.2. Images on s390x (IBM Z) architecture

OS	Java	Red Hat Ecosystem Catalog
RHEL 8	Eclipse OpenJ9 11	RHEL 8 with Eclipse OpenJ9 11

3.2.1.3. Images on ppc64le (IBM Power Systems) architecture

OS	Java	Red Hat Ecosystem Catalog
RHEL 8	Eclipse OpenJ9 11	RHEL 8 with Eclipse OpenJ9 11

NOTE

The use of a RHEL 8-based container on a RHEL 7 host, for example with OpenShift 3 or OpenShift 4, has limited support. For more information, see the Red Hat Enterprise Linux Container Compatibility Matrix.

3.2.2. Preparing Thorntail application for OpenShift deployment

For deploying your Thorntail application to OpenShift, it must contain:

- Launcher profile information in the application's **pom.xml** file.

In the following procedure, a profile with OpenShift Maven plugin is used for building and deploying the application to OpenShift.

Prerequisites

- Maven is installed.
- Docker or podman authentication into Red Hat Ecosystem Catalog to access RHEL 8 images.

Procedure

1. Add the following content to the **pom.xml** file in the application root directory:

```
...

<profiles>
  <profile>
   <id>openshift</id>
   <build>
    <plugins>
     <plugin>
```

```
          <groupId>org.eclipse.jkube</groupId>
          <artifactId>openshift-maven-plugin</artifactId>
          <version>1.1.0</version>
          <executions>
            <execution>
              <goals>
                <goal>resource</goal>
                <goal>build</goal>
              </goals>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
  </profile>
</profiles>
```

2. Replace the **jkube.generator.from** property in the **pom.xml** file to specify the OpenJDK image that you want to use.

   - x86_64 architecture
     - RHEL 7 with OpenJDK 8

       ```
       <jkube.generator.from>registry.access.redhat.com/redhat-openjdk-18/openjdk18-openshift:latest</jkube.generator.from>
       ```

     - RHEL 7 with OpenJDK 11

       ```
       <jkube.generator.from>registry.access.redhat.com/openjdk/openjdk-11-rhel7:latest</jkube.generator.from>
       ```

     - RHEL 8 with OpenJDK 8

       ```
       <jkube.generator.from>registry.redhat.io/openjdk/openjdk-8-rhel8:latest</jkube.generator.from>
       ```

     - RHEL 8 with OpenJDK 11

       ```
       <jkube.generator.from>registry.redhat.io/openjdk/openjdk-11-rhel8:latest</jkube.generator.from>
       ```

   - s390x (IBM Z) architecture
     - RHEL 8 with Eclipse OpenJ9 11

       ```
       <jkube.generator.from>registry.access.redhat.com/openj9/openj9-11-rhel8:latest</jkube.generator.from>
       ```

   - ppc64le (IBM Power Systems) architecture
     - RHEL 8 with Eclipse OpenJ9 11

```
<fabric8.generator.from>registry.access.redhat.com/openj9/openj9-11-
rhel8:latest</fabric8.generator.from>
```

3.2.3. Deploying Thorntail application to OpenShift using OpenShift Maven plugin

To deploy your Thorntail application to OpenShift, you must perform the following:

- Log in to your OpenShift instance.
- Deploy the application to the OpenShift instance.

Prerequisites

- **oc** CLI client installed.
- Maven installed.

Procedure

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

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

2. Create a new project in the OpenShift instance.

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

3. Deploy the application to OpenShift using Maven from the application's root directory. The root directory of an application contains the **pom.xml** file.

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

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

4. Verify the deployment.

   a. Check the status of your application 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.

   b. Determine the route for the pod.

   #### 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 this example, **http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME** is the base URL to access the application.

c. Verify that your application is running in OpenShift.

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

3.3. DEPLOYING THORNTAIL APPLICATION TO STAND-ALONE RED HAT ENTERPRISE LINUX

To deploy your Thorntail application to stand-alone Red Hat Enterprise Linux, configure the **pom.xml** file in the application, package it using Maven and deploy using the **java -jar** command.

Prerequisites

- RHEL 7 or RHEL 8 installed.

3.3.1. Preparing Thorntail application for stand-alone Red Hat Enterprise Linux deployment

For deploying your Thorntail application to stand-alone Red Hat Enterprise Linux, you must first package the application using Maven.

Prerequisites

- Maven installed.

Procedure

1. Add the following content to the **pom.xml** file in the application’s root directory:

```
...
<build>
  <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>
...
```

2. Package your application using Maven.

```
$ mvn clean package
```

The resulting JAR file is in the **target** directory.

3.3.2. Deploying Thorntail application to stand-alone Red Hat Enterprise Linux using jar

To deploy your Thorntail application to stand-alone Red Hat Enterprise Linux, use **java -jar** command.

Prerequisites

- RHEL 7 or RHEL 8 installed.
- OpenJDK 8 or OpenJDK 11 installed.
- A JAR file with the application.

Procedure

1. Deploy the JAR file with the application.

```
$ java -jar my-app-thorntail.jar
```

2. Verify the deployment.
Use **curl** or your browser to verify your application is running at **http://localhost:8080**:

```
$ curl http://localhost:8080
```

CHAPTER 4. USING THORNTAIL MAVEN PLUGIN

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

4.1. 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>
```

4.2. THORNTAIL MAVEN PLUGIN GOALS

The Thorntail Maven plugin provides several goals:

package

Creates the executable package (see Section 9.2, “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. THORNTAIL MAVEN PLUGIN CONFIGURATION OPTIONS

The Thorntail Maven plugin accepts the following configuration options:

bundleDependencies

If **true**, dependencies are included in the **-thorntail.jar** file. Otherwise, they are resolved from local Maven repository or from the network at runtime.

Property	**thorntail.bundleDependencies**
Default	true
Used by	**package**

NOTE

In case you set **bundleDependencies** to **false**, a local Maven repository is used at runtime. The location of the Maven repository is retrieved either from a **settings.xml** file, or from a system property.

1. If a **settings.xml** file is found and it defines a local repository, then this repository is used. The following locations are searched for the **settings.xml** file. The first occurrence of the file is used to retrieve the location of the Maven repository:
   a. If the **jboss.modules.settings.xml.url** system property is defined, then the **settings.xml** file defined in this property is used. The value must be a **file://...** URL .
   b. If the ~/**.m2/settings.xml** file exists, then it is used.
   c. If the **$M2_HOME/conf/settings.xml** file exists, then it is used.
2. If the path to the Maven repository is not defined in a **settings.xml** file, then:
   a. If the **maven.repo.local** system property is defined, then this repository is used.
   b. Otherwise the standard location ~/**.m2/repository** is used.

debug

The port to use for debugging. If set, the Thorntail process suspends on start and opens a debugger on this port.

Property	**thorntail.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	**thorntail.environmentFile**
Default	
Used by	**multistart**, **run**, **start**

filterWebInfLib

If **true**, the plugin removes artifacts that are provided by the Thorntail runtime from the **WEB-INF/lib** directory of the project WAR file. If **uberjar-only**, this transformation is only done on the WAR file that is put into the uberjar, and the standalone WAR file remains untouched. Otherwise, the contents of **WEB-INF/lib** remain untouched.

Property	**thorntail.filterWebInfLib**
Default	true
Used by	**package**

NOTE

This option is generally not necessary and is provided as a workaround in case the Thorntail plugin removes a dependency required by the application. When false, it is the responsibility of the developer to ensure that the **WEB-INF/lib** directory does not contain Thorntail artifacts that would compromise the functionality of the application. One way to do that is to avoid expressing dependencies on fractions and rely on auto-detection or by explicitly listing any required extra fractions using the **fractions** option.

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	**thorntail.detect.mode**

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

fractions

A list of extra fractions to include when using auto-detection. It is useful for fractions that cannot be detected or for user-provided fractions.
Use one of the following formats when specifying a fraction: * **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 a **!** character, the corresponding auto-detected fraction is not installed (unless it is a dependency of any other fraction). In the following example the Undertow fraction is not installed even though 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	**thorntail.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 4.4, “Thorntail Maven plugin configuration properties” .

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

propertiesFile

See Section 4.4, “Thorntail Maven plugin configuration properties” .

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

stderrFile

Specifies the path to a file where the **stderr** output is stored instead of being sent to the **stderr** output of the launching process.

Property	**thorntail.stderr**
Default	
Used by	**run**, **start**

stdoutFile

Specifies the path to a file where the **stdout** output is stored instead of being sent to the **stdout** output of the launching process.

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

useUberJar

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

Property	**thorntail.useUberJar**
Default	
Used by	**run**, **start**

4.4. 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.**

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

CHAPTER 5. USING THORNTAIL FRACTIONS

5.1. 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 9.2, “Creating an uberjar”).

To enable easier consumption of Thorntail fractions, a bill of materials (BOM) is available. For more information, see Chapter 6, *Using a BOM*.

5.2. 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_2.1_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 5.1, “Fractions”
- Section 9.2, “Creating an uberjar”

5.3. 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 Chapter 6, *Using a BOM*.
2. Add the Thorntail Maven plugin to your **pom.xml**. For more information, see Section 9.2, “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 5.1, “Fractions”
- Section 5.2, “Auto-detecting fractions”

CHAPTER 6. 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.

6.1. 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.

6.2. 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>2.7.3.Final-redhat-00001</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**.

CHAPTER 7. ACCESSING LOGS ON YOUR THORNTAIL APPLICATION

7.1. 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 **thorntail.logging** system property:

   ```
   $ mvn thorntail:run -Dthorntail.logging=FINE
   ```

   See the **org.wildfly.swarm.config.logging.Level** class for the list of available levels.

7.2. 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 7.1, “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:

```
thorntail:
  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

```
thorntail:
  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
```

CHAPTER 8. 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.

8.1. SYSTEM PROPERTIES

Using system properties for configuring your application is advantageous for experimenting, debugging, and other short-term activities.

8.1.1. Commonly used system properties

This is a non-exhaustive list of system properties you are likely to use in your application:

General system properties

thorntail.bind.address

The interface to bind servers

Default	0.0.0.0

thorntail.port.offset

The global port adjustment

Default	0

thorntail.context.path

The context path for the deployed application

Default	/

thorntail.http.port

The port for the HTTP server

Default	8080

thorntail.https.port

The port for the HTTPS server

Default	8443

thorntail.debug.port

If provided, the Thorntail process will pause for debugging on the given port.

This option is only available when running an Arquillian test or starting the application using the **mvn thorntail:run** command, not when executing a JAR file. The JAR file execution requires normal Java debug agent parameters.

Default	

thorntail.io.tmpdir

Temporary directory Thorntail uses for unpacking the uberjar and for other temporary files.

Default	**java.io.tmpdir**

Datasource-related system properties

With JDBC driver autodetection, use the following properties to configure the datasource:

thorntail.ds.name

The name of the datasource

Default	ExampleDS

thorntail.ds.username

The user name to access the database

Default	*driver-specific*

thorntail.ds.password

The password to access the database

Default	*driver-specific*

thorntail.ds.connection.url

The JDBC connection URL

Default	*driver-specific*

NOTE

For a full set of available properties, see the documentation for each fraction and the javadocs on class SwarmProperties.java

8.1.2. 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 **thorntail.undertow.servers.KEY.default-host** indicates that the configuration applies to a particular named server.

In practical usage, the property would be, for example, **thorntail.undertow.servers.default.default-host** for a server known as **default**.

8.1.3. 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>2.7.3.Final-redhat-00001</version>
      <configuration>
        <properties>
          <thorntail.bind.address>127.0.0.1</thorntail.bind.address>
          <java.net.preferIPv4Stack>true</java.net.preferIPv4Stack>
        </properties>
      </configuration>
    </plugin>
  </plugins>
</build>
```

In the example above, the **thorntail.bind.address** property is set to **127.0.0.1** and the **java.net.preferIPv4Stack** property is set to **true**.

8.1.4. 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 -Dthorntail.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 **thorntail.bind.address**.

8.1.5. Specifying external JDBC drivers

When executing an application, either as an uber JAR or using a hollow JAR, you can specify a JDBC driver JAR using the **thorntail.classpath** system property. This way, you do not need to package the driver in the application.

The **thorntail.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 -Dthorntail.classpath=./h2-1.4.196.jar -jar microprofile-jpa-hollow-thorntail.jar example-jpa-jaxrs-cdi.war
   ```

8.2. ENVIRONMENT VARIABLES

Use environment variables to configure your application or override values stored in YAML files.

8.2.1. Application configuration using environment variables

Use environment variables to configure your application in various deployments—especially in a containerized environment, such as Docker.

Example 8.1. Environment variables configuration

A property documented as **thorntail.undertow.servers.KEY.default-host** translates to the following environment variable (substituting the **KEY** segment with the **default** identifier):

```
export THORNTAIL.UNDERTOW.SERVERS.DEFAULT.DEFAULT_DASH_HOST=<myhost>
```

Unlike other configuration options, properties defined as environment variables in Linux-based containers do not allow defining non-alphanumeric characters like *dot* (.), *dash/hyphen* (-) or any other characters not in the **[A-Za-z0-9_]** range. Many configuration properties in Thorntail contain these characters, so you must follow these rules when defining the environment variables in the following environments:

Linux-based container rules

- It is a naming convention that all environment properties are defined using uppercase letters. For example, define the **serveraddress** property as **SERVERADDRESS**.
- All the *dot* (.) characters must be replaced with *underscore* (_). For example, define the **thorntail.bind.address=127.0.0.1** property as **THORNTAIL_BIND_ADDRESS=127.0.0.1**.
- All *dash/hyphen* (-) characters must be replaced with the **_DASH_** string. For example, define the **thorntail.data-sources.foo.url=<url>** property as **THORNTAIL_DATA_DASH_SOURCES_FOO_URL=<url>**.
- If the property name contains underscores, all *underscore* (_) characters must be replaced with the **_UNDERSCORE_** string. For example, define the **thorntail.data_sources.foo.url=<url>** property as **THORNTAIL_DATA_UNDERSCORE_SOURCES_FOO_URL=<url>**.

Example 8.2. An example data source configuration

System property	**-Dthorntail.datasources.data-sources.devwf.connection-url= jdbc:postgresql://localhost:5432/sampledb**
Env. variable	**THORNTAIL_DATASOURCES_DATA_DASH_SOURCES_DEVWF_CONNECTION_DASH_URL= 'jdbc:postgresql://localhost:5432/sampledb'**

System property	**-Dthorntail.datasources.data-sources.devwf.driver-name=postgresql**
Env. variable	**THORNTAIL_DATASOURCES_DATA_DASH_SOURCES_DEVWF_DRIVER_DASH_NAME='postgresql'**

System property	**-Dthorntail.datasources.data-sources.devwf.jndiname=java:/jboss/datasources/devwf**
Env. variable	**THORNTAIL_DATASOURCES_DATA_DASH_SOURCES_DEVWF_JNDI_DASH_NAME='java:/jboss/datasources/devwf'**

System property	**-Dthorntail.datasources.data-sources.devwf.user-name=postgres**
Env. variable	**THORNTAIL_DATASOURCES_DATA_DASH_SOURCES_DEVWF_USER_DASH_NAME='postgres'**

System property	**-Dthorntail.datasources.data-sources.devwf.password=admin**
Env. variable	**THORNTAIL_DATASOURCES_DATA_DASH_SOURCES_DEVWF_PASSWORD='admin'**

8.3. 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.

8.3.1. The general YAML file format

The Thorntail configuration item names correspond to the YAML configuration structure. That is, if you want to write a piece of YAML configuration for some configuration property, you just need to separate the configuration property around the **.** characters.

Example 8.3. YAML configuration

For example, a configuration item documented as **thorntail.undertow.servers.KEY.default-host** translates to the following YAML structure, substituting the **KEY** segment with the **default** identifier:

```
thorntail:
  undertow:
    servers:
      default:
        default-host: <myhost>
```

This simple rule applies always, there are no exceptions and no additional delimiters. Most notably, some Eclipse MicroProfile specifications define configuration properties that use / as a delimiter, because the **.** character is used in fully qualified class names. When writing the YAML configuration, it is still required to split around **.** and *not* around **/**.

Example 8.4. YAML configuration for MicroProfile Rest Client

For example, MicroProfile Rest Client specifies that you can configure URL of an external service with a configuration property named **com.example.demo.client.Service/mp-rest/url**. This translates to the following YAML:

```
com:
```

```
example:
  demo:
    client:
      Service/mp-rest/url: http://localhost:8080/...
```

8.3.2. 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 8.5. 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**

8.3.3. 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 8.6. 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 8.3.2, “Default Thorntail YAML Files”

CHAPTER 9. PACKAGING YOUR APPLICATION

This sections contains information about packaging your Thorntail–based application for deployment and execution.

9.1. 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:

9.1.1. 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
```

9.1.2. 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 **thorntail.classpath** system property, for example:
>
> ```
> $ java -Dthorntail.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
```

9.1.2.1. 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>
```

9.2. 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 10. TESTING YOUR APPLICATION

10.1. 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 Chapter 6, *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.

```
thorntail:
  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-module-name: com.h2database.h2
      driver-xa-datasource-class-name: org.h2.jdbcx.JdbcDataSource
```

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 11. DEBUGGING YOUR APPLICATION

This sections contains information about debugging your Thorntail–based application both in local and remote deployments.

11.1. 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.

11.1.1. 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 **-Dthorntail.debug.port** argument:

   ```
   $ mvn thorntail:run -Dthorntail.debug.port=$PORT_NUMBER
   ```

   Here, **$PORT_NUMBER** is an unused port number of your choice. Remember this number for the remote debugger configuration.

11.1.2. 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_NUMBER -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 9.2, "Creating an uberjar"

11.1.3. 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.
- 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**.

11.1.4. 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 CodeReady 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 CodeReady Studio installed on your machine. You can download it from the Red Hat CodeReady Studio download page.

Procedure

1. Start Red Hat CodeReady 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.
  Red Hat CodeReady Studio was previously called JBoss Developer Studio.

- A Debugging Java Applications On OpenShift and Kubernetes article on OpenShift Blog.

11.2. DEBUG LOGGING

11.2.1. Local debug logging

To enable debug logging locally, see the Section 7.1, "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
```

11.2.2. Accessing debug logs on OpenShift

Start your application and interact with it to see the debugging statements in OpenShift.

Prerequisites

- The **oc** CLI client installed and authenticated.

- A Maven-based application with debug logging enabled.

Procedure

1. Deploy your application to OpenShift:

   ```
   $ mvn clean oc: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 example 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 application.

Additional resources

- Section D.24, “Logging”

CHAPTER 12. MONITORING YOUR APPLICATION

This section contains information about monitoring your Thorntail–based application running on OpenShift.

12.1. ACCESSING JVM METRICS FOR YOUR APPLICATION ON OPENSHIFT

12.1.1. 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

12.2. APPLICATION METRICS

Thorntail provides ways of exposing application metrics in order to track performance and service availability.

12.2.1. What are metrics

In the microservices architecture, where multiple services are invoked in order to serve a single user request, diagnosing performance issues or reacting to service outages might be hard. To make solving problems easier, applications must expose machine-readable data about their behavior, such as:

- How many requests are currently being processed.

- How many connections to the database are currently in use.

- How long service invocations take.

These kinds of data are referred to as *metrics*. Collecting metrics, visualizing them, setting alerts, discovering trends, etc. are very important to keep a service healthy.

Thorntail provides a fraction for Eclipse MicroProfile Metrics, an easy-to-use API for exposing metrics. Among other formats, it supports exporting data in the native format of Prometheus, a popular monitoring solution. Inside the application, you need nothing except this fraction. Outside of the application, Prometheus typically runs.

Additional resources

- The MicroProfile Metrics GitHub page .

- The Prometheus homepage

- A popular solution to visualize metrics stored in Prometheus is Grafana. For more information, see the Grafana homepage.

12.2.2. Exposing application metrics

In this example, you:

- Configure your application to expose metrics.
- Collect and view the data using Prometheus.

Note that Prometheus actively connects to a monitored application to collect data; the application does not actively send metrics to a server.

Prerequisites

- Prometheus configured to collect metrics from the application:
  1. Download and extract the archive with the latest Prometheus release:

     ```
     $ wget
     https://github.com/prometheus/prometheus/releases/download/v2.4.3/prometheus-
     2.4.3.linux-amd64.tar.gz
     $ tar -xvf  prometheus-2.4.3.linux-amd64.tar.gz
     ```

  2. Navigate to the directory with Prometheus:

     ```
     $ cd  prometheus-2.4.3.linux-amd64
     ```

  3. Append the following snippet to the **prometheus.yml** file to make Prometheus automatically collect metrics from your application:

     ```
       - job_name: 'thorntail'
         static_configs:
         - targets: ['localhost:8080']
     ```

     The default behavior of Thorntail-based applications is to expose metrics at the **/metrics** endpoint. This is what the MicroProfile Metrics specification requires, and also what Prometheus expects.

- The Prometheus server started on **localhost**:
  Start Prometheus and wait until the **Server is ready to receive web requests** message is displayed in the console.

  ```
  $ ./prometheus
  ```

Procedure

1. Include the **microprofile-metrics** fraction in the **pom.xml** file in your application:

   pom.xml

   ```
   <dependencies>
     <dependency>
       <groupId>io.thorntail</groupId>
       <artifactId>microprofile-metrics</artifactId>
     </dependency>
   </dependencies>
   ```

2. Annotate methods or classes with the metrics annotations, for example:

```
@GET
@Counted(name = "hello-count", absolute = true)
@Timed(name = "hello-time", absolute = true)
public String get() {
    return "Hello from counted and timed endpoint";
}
```

Here, the **@Counted** annotation is used to keep track of how many times this method was invoked. The **@Timed** annotation is used to keep track of how long the invocations took.

In this example, a JAX-RS resource method was annotated directly, but you can annotate any CDI bean in your application as well.

3. Launch your application:

   ```
   $ mvn thorntail:run
   ```

4. Invoke the traced endpoint several times:

   ```
   $ curl http://localhost:8080/
   Hello from counted and timed endpoint
   ```

5. Wait at least 15 seconds for the collection to happen, and see the metrics in Prometheus UI:

   1. Open the Prometheus UI at http://localhost:9090/ and type **hello** into the *Expression* box.

   2. From the suggestions, select for example **application:hello_count** and click *Execute*.

   3. In the table that is displayed, you can see how many times the resource method was invoked.

   4. Alternatively, select **application:hello_time_mean_seconds** to see the mean time of all the invocations.

   Note that all metrics you created are prefixed with **application:**. There are other metrics, automatically exposed by Thorntail as the MicroProfile Metrics specification requires. Those metrics are prefixed with **base:** and **vendor:** and expose information about the JVM in which the application runs.

Additional resources

- For additional types of metrics, see the Eclipse MicroProfile Metrics documentation .

CHAPTER 13. AVAILABLE EXAMPLES FOR THORNTAIL

The Thorntail runtime provides example applications. When you start developing applications on OpenShift, you can use the example applications as templates.

You can access these example applications on Developer Launcher.

You can download and deploy all the example applications on:

- x86_64 architecture - The example applications in this guide demonstrate how to build and deploy example applications on x86_64 architecture.
- s390x architecture - To deploy the example applications on OpenShift environments provisioned on IBM Z infrastructure, specify the relevant IBM Z image name in the commands.
- ppc64le architecture - To deploy the example applications on OpenShift environments provisioned on IBM Power Systems infrastructure, specify the relevant IBM Power Systems image name in the commands.

Refer to the section Supported Java images for Thorntail for more information about the image names.

Some of the example applications also require other products, such as Red Hat Data Grid to demonstrate the workflows. In this case, you must also change the image names of these products to their relevant IBM Z and IBM Power Systems image names in the YAML file of the example applications.

13.1. REST API LEVEL 0 EXAMPLE FOR THORNTAIL

IMPORTANT

The following example is not meant to be run in a production environment.

Example proficiency level: **Foundational**.

What the REST API Level 0 example does

The REST API Level 0 example 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 example 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.

13.1.1. REST API Level 0 design tradeoffs

Table 13.1. Design tradeoffs

Pros	Cons
● The example application 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.

13.1.2. Deploying the REST API Level 0 example application to OpenShift Online

Use one of the following options to execute the REST API Level 0 example application 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 deployment workflow that executes the **oc** commands for you.

13.1.2.1. Deploying the example application using developers.redhat.com/launch

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Red Hat Developer Launcher web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

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

13.1.2.2. Authenticating the oc CLI client

To work with example applications on OpenShift Online using the **oc** command-line client, you must 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

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

13.1.2.3. Deploying the REST API Level 0 example application using the oc CLI client

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using developers.redhat.com/launch. For more information, see Section 13.1.2.1, "Deploying the example application using developers.redhat.com/launch" .
- The **oc** client authenticated. For more information, see Section 13.1.2.2, "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 application.
4. Use Maven to start the deployment to OpenShift.

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

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

5. Check the status of your application 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. After your example application 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.

13.1.3. Deploying the REST API Level 0 example application to Minishift or CDK

Use one of the following options to execute the REST API Level 0 example application locally on Minishift or CDK:

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

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

13.1.3.1. Getting the Launcher tool URL and credentials

You need the Launcher tool URL and user credentials to create and deploy example applications on Minishift or CDK. This information is provided when the Minishift or CDK is started.

Prerequisites

- The Launcher tool installed, configured, and running.

Procedure

1. Navigate to the console where you started Minishift or CDK.
2. Check the console output for the URL and user credentials you can use to access the running Launcher:

   Example Console Output from a Minishift or CDK 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
```

13.1.3.2. Deploying the example application using the Launcher tool

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Launcher web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.1.3.1, “Getting the Launcher tool URL and credentials” .

Procedure

1. Navigate to the Launcher URL in a browser.

2. Follow the on-screen instructions to create and launch your example application in Thorntail.

13.1.3.3. Authenticating the oc CLI client

To work with example applications on Minishift or CDK using the **oc** command-line client, you must authenticate the client using the token provided by the Minishift or CDK web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.1.3.1, “Getting the Launcher tool URL and credentials” .

Procedure

1. Navigate to the Minishift or CDK 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. Copy the **oc login** command.

5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your Minishift or CDK account.

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

13.1.3.4. Deploying the REST API Level 0 example application using the oc CLI client

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using Launcher tool on a Minishift or CDK. For more information, see Section 13.1.3.2, "Deploying the example application using the Launcher tool" .
- Your Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 13.1.3.3, "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 application.
4. Use Maven to start the deployment to OpenShift.

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

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

5. Check the status of your application 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. After your example application 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.

13.1.4. Deploying the REST API Level 0 example application to OpenShift Container Platform

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

Prerequisites

- The example application created using developers.redhat.com/launch.

Procedure

- Follow the instructions in Section 13.1.2, “Deploying the REST API Level 0 example application to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

13.1.5. Interacting with the unmodified REST API Level 0 example application for Thorntail

The example 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 example. 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 example. 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 example to perform these same interactions. The form is located at the root of the project **http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME**.

13.1.6. Running the REST API Level 0 example application integration tests

This example application 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 example application from the target OpenShift project. To avoid accidentally removing your example 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
```

13.1.7. 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 for Spring Boot

- REST API Level 0 for Eclipse Vert.x
- REST API Level 0 for Node.js

13.2. EXTERNALIZED CONFIGURATION EXAMPLE FOR THORNTAIL

IMPORTANT

The following example is not meant to be run in a production environment.

Example proficiency level: **Foundational**.

Externalized Configuration 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 example 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.

13.2.1. 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.

13.2.2. Externalized Configuration design tradeoffs

Table 13.2. Design Tradeoffs

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

13.2.3. Deploying the Externalized Configuration example application to OpenShift Online

Use one of the following options to execute the Externalized Configuration example application 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 deployment workflow that executes the **oc** commands for you.

13.2.3.1. Deploying the example application using developers.redhat.com/launch

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Red Hat Developer Launcher web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

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

13.2.3.2. Authenticating the oc CLI client

To work with example applications on OpenShift Online using the **oc** command-line client, you must 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. Copy the **oc login** command.

5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

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

13.2.3.3. Deploying the Externalized Configuration example application using the oc CLI client

This section shows you how to build your Externalized Configuration example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using developers.redhat.com/launch. For more information, see Section 13.2.3.1, "Deploying the example application using developers.redhat.com/launch" .

- The **oc** client authenticated. For more information, see Section 13.2.3.2, "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 application.

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

   ```
   $ 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: template.openshift.io/v1
   data:
   ```

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

6. Use Maven to start the deployment to OpenShift.

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

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

7. Check the status of your application 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. After your example application 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.

13.2.4. Deploying the Externalized Configuration example application to Minishift or CDK

Use one of the following options to execute the Externalized Configuration example application locally on Minishift or CDK:

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

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

13.2.4.1. Getting the Launcher tool URL and credentials

You need the Launcher tool URL and user credentials to create and deploy example applications on Minishift or CDK. This information is provided when the Minishift or CDK is started.

Prerequisites

- The Launcher tool installed, configured, and running.

Procedure

1. Navigate to the console where you started Minishift or CDK.
2. Check the console output for the URL and user credentials you can use to access the running Launcher:

   Example Console Output from a Minishift or CDK 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
```

13.2.4.2. Deploying the example application using the Launcher tool

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Launcher web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.2.4.1, “Getting the Launcher tool URL and credentials” .

Procedure

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

13.2.4.3. Authenticating the oc CLI client

To work with example applications on Minishift or CDK using the **oc** command-line client, you must authenticate the client using the token provided by the Minishift or CDK web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.2.4.1, “Getting the Launcher tool URL and credentials” .

Procedure

1. Navigate to the Minishift or CDK 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your Minishift or CDK account.

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

13.2.4.4. Deploying the Externalized Configuration example application using the oc CLI client

This section shows you how to build your Externalized Configuration example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using Launcher tool on a Minishift or CDK. For more information, see Section 13.2.4.2, "Deploying the example application using the Launcher tool" .
- Your Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 13.2.4.3, "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 application.
4. Deploy your ConfigMap configuration to OpenShift using **app-config.yml** in the root of the example.

   ```
   $ 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: template.openshift.io/v1
data:
  app-config.yml: |-
    greeting:
      message: Hello %s from a ConfigMap!
...
```

6. Use Maven to start the deployment to OpenShift.

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

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

7. Check the status of your application 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. After your example application 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.

13.2.5. Deploying the Externalized Configuration example application to OpenShift Container Platform

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

Prerequisites

- The example application created using developers.redhat.com/launch.

Procedure

- Follow the instructions in Section 13.2.3, “Deploying the Externalized Configuration example application to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

13.2.6. Interacting with the unmodified Externalized Configuration example application for Thorntail

The example 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 example. 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 example 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 example 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!"}
   ```

13.2.7. Running the Externalized Configuration example application integration tests

This example application 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 example application from the target OpenShift project. To avoid accidentally removing your example 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
```

13.2.8. 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 for Spring Boot
- Externalized Configuration for Eclipse Vert.x
- Externalized Configuration for Node.js

13.3. RELATIONAL DATABASE BACKEND EXAMPLE FOR THORNTAIL

IMPORTANT

The following example is not meant to be run in a production environment.

Limitation: Run this example application on a Minishift or CDK. You can also use a manual workflow to deploy this example to OpenShift Online Pro and OpenShift Container Platform. This example is not currently available on OpenShift Online Starter.

Example proficiency level: **Foundational**.

What the Relational Database Backend example does

The Relational Database Backend example expands on the REST API Level 0 application 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 example 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 example 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 a user interface provided by the example to use the application. Specifically, this example 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 example also contains a set of automated integration tests that can be used to verify that the application is fully integrated with the database.

This example 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.

13.3.1. Relational Database Backend design tradeoffs

Table 13.3. 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 provided with this example application 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 example application's pod in order to preserve changes, see the Creating an application with a database 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).

13.3.2. Deploying the Relational Database Backend example application to OpenShift Online

Use one of the following options to execute the Relational Database Backend example application 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 deployment workflow that executes the **oc** commands for you.

13.3.2.1. Deploying the example application using developers.redhat.com/launch

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Red Hat Developer Launcher web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

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

13.3.2.2. Authenticating the oc CLI client

To work with example applications on OpenShift Online using the **oc** command-line client, you must 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

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

13.3.2.3. Deploying the Relational Database Backend example application using the oc CLI client

This section shows you how to build your Relational Database Backend example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using developers.redhat.com/launch. For more information, see Section 13.3.2.1, "Deploying the example application using developers.redhat.com/launch" .
- The **oc** client authenticated. For more information, see Section 13.3.2.2, "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 application.
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 example application is pre-configured to use these values. Using different values prevents your application from integrating with the database.

```
$ oc new-app -e POSTGRESQL_USER=luke -ePOSTGRESQL_PASSWORD=secret -
ePOSTGRESQL_DATABASE=my_data registry.access.redhat.com/rhscl/postgresql-10-rhel7
--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 oc:deploy -Popenshift
   ```

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

7. Check the status of your application 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. After your example application 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.

13.3.3. Deploying the Relational Database Backend example application to Minishift or CDK

Use one of the following options to execute the Relational Database Backend example application locally on Minishift or CDK:

- Using Launcher

- Using the **oc** CLI client

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

13.3.3.1. Getting the Launcher tool URL and credentials

You need the Launcher tool URL and user credentials to create and deploy example applications on Minishift or CDK. This information is provided when the Minishift or CDK is started.

Prerequisites

- The Launcher tool installed, configured, and running.

Procedure

1. Navigate to the console where you started Minishift or CDK.

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

   Example Console Output from a Minishift or CDK 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
```

13.3.3.2. Deploying the example application using the Launcher tool

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Launcher web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.3.3.1, “Getting the Launcher tool URL and credentials” .

Procedure

1. Navigate to the Launcher URL in a browser.

2. Follow the on-screen instructions to create and launch your example application in Thorntail.

13.3.3.3. Authenticating the oc CLI client

To work with example applications on Minishift or CDK using the **oc** command-line client, you must authenticate the client using the token provided by the Minishift or CDK web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.3.3.1, “Getting the Launcher tool URL and credentials” .

Procedure

1. Navigate to the Minishift or CDK 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your Minishift or CDK account.

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

13.3.3.4. Deploying the Relational Database Backend example application using the oc CLI client

This section shows you how to build your Relational Database Backend example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using Launcher tool on a Minishift or CDK. For more information, see Section 13.3.3.2, “Deploying the example application using the Launcher tool” .
- Your Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 13.3.3.3, “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 application.

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 example application is pre-configured to use these values. Using different values prevents your application from integrating with the database.

```
$ oc new-app -e POSTGRESQL_USER=luke -ePOSTGRESQL_PASSWORD=secret -
ePOSTGRESQL_DATABASE=my_data registry.access.redhat.com/rhscl/postgresql-10-rhel7
--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 oc:deploy -Popenshift
```

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

7. Check the status of your application 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. After your example application 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.

13.3.4. Deploying the Relational Database Backend example application to OpenShift Container Platform

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

Prerequisites

- The example application created using developers.redhat.com/launch.

Procedure

- Follow the instructions in Section 13.3.2, “Deploying the Relational Database Backend example application to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

13.3.5. Interacting with the Relational Database Backend API

When you have finished creating your example application, 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" : "Apple",
  "stock" : 10
}, {
  "id" : 2,
  "name" : "Orange",
  "stock" : 10
}, {
  "id" : 3,
  "name" : "Pear",
  "stock" : 10
} ]
```

Retrieve an entry with a specific ID

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

```
{
  "id" : 3,
  "name" : "Pear",
  "stock" : 10
}
```

Create a new entry:

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

```
{
  "id" : 4,
  "name" : "Peach",
  "stock" : 1
}
```

Update an Entry

```
$ curl -H "Content-Type: application/json" -X PUT -d '{"name":"Apple","stock":100}'
http://MY_APP_NAME-MY_PROJECT_NAME.OPENSHIFT_HOSTNAME/api/fruits/1
```

```
{
  "id" : 1,
  "name" : "Apple",
  "stock" : 100
}
```

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.

13.3.6. Running the Relational Database Backend example application integration tests

This example application 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 example application from the target OpenShift project. To avoid accidentally removing your example 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
```

13.3.7. 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 for Spring Boot
- Relational Database Backend for Eclipse Vert.x
- Relational Database Backend for Node.js

13.4. HEALTH CHECK EXAMPLE FOR THORNTAIL

IMPORTANT

The following example is not meant to be run in a production environment.

Example proficiency level: **Foundational**.

When you deploy an application, it is 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 13.4.1, “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.

13.4.1. 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.

13.4.2. Deploying the Health Check example application to OpenShift Online

Use one of the following options to execute the Health Check example application 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 deployment workflow that executes the **oc** commands for you.

13.4.2.1. Deploying the example application using developers.redhat.com/launch

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Red Hat Developer Launcher web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

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

13.4.2.2. Authenticating the oc CLI client

To work with example applications on OpenShift Online using the **oc** command-line client, you must 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. Copy the **oc login** command.

5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

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

13.4.2.3. Deploying the Health Check example application using the oc CLI client

This section shows you how to build your Health Check example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using developers.redhat.com/launch. For more information, see Section 13.4.2.1, "Deploying the example application using developers.redhat.com/launch" .

- The **oc** client authenticated. For more information, see Section 13.4.2.2, "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 application.

4. Use Maven to start the deployment to OpenShift.

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

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

5. Check the status of your application 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. After your example application 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.

13.4.3. Deploying the Health Check example application to Minishift or CDK

Use one of the following options to execute the Health Check example application locally on Minishift or CDK:

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

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

13.4.3.1. Getting the Launcher tool URL and credentials

You need the Launcher tool URL and user credentials to create and deploy example applications on Minishift or CDK. This information is provided when the Minishift or CDK is started.

Prerequisites

- The Launcher tool installed, configured, and running.

Procedure

1. Navigate to the console where you started Minishift or CDK.
2. Check the console output for the URL and user credentials you can use to access the running Launcher:

   Example Console Output from a Minishift or CDK 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
```

13.4.3.2. Deploying the example application using the Launcher tool

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Launcher web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.4.3.1, "Getting the Launcher tool URL and credentials" .

Procedure

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

13.4.3.3. Authenticating the oc CLI client

To work with example applications on Minishift or CDK using the **oc** command-line client, you must authenticate the client using the token provided by the Minishift or CDK web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.4.3.1, "Getting the Launcher tool URL and credentials" .

Procedure

1. Navigate to the Minishift or CDK 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your Minishift or CDK account.

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

13.4.3.4. Deploying the Health Check example application using the oc CLI client

This section shows you how to build your Health Check example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using Launcher tool on a Minishift or CDK. For more information, see Section 13.4.3.2, “Deploying the example application using the Launcher tool” .
- Your Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 13.4.3.3, “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 application.

4. Use Maven to start the deployment to OpenShift.

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

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

5. Check the status of your application 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. After your example application 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.

13.4.4. Deploying the Health Check example application to OpenShift Container Platform

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

Prerequisites

- The example application created using developers.redhat.com/launch.

Procedure

- Follow the instructions in Section 13.4.2, "Deploying the Health Check example application to OpenShift Online", only use the URL and user credentials from the OpenShift Container Platform Web Console.

13.4.5. Interacting with the unmodified Health Check example application

After you deploy the example application, you will have the **MY_APP_NAME** service running. The **MY_APP_NAME** service 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 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.

Message	Status
Pulling	Downloading the latest version of docker image to re-create the container.
Pulled	Docker image downloaded successfully.
Created	Docker container has been successfully created
Started	Docker container is ready to handle requests

13.4.6. Running the Health Check example application integration tests

This example application 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 example application from the target OpenShift project. To avoid accidentally removing your example 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
```

13.4.7. Health check resources

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

- Application Health in OpenShift
- Kubernetes Liveness and Readiness Probes

- Health Check for Spring Boot
- Health Check for Eclipse Vert.x
- Health Check for Node.js

13.5. CIRCUIT BREAKER EXAMPLE FOR THORNTAIL

IMPORTANT

The following example is not meant to be run in a production environment.

Limitation: Run this example application on a Minishift or CDK. You can also use a manual workflow to deploy this example to OpenShift Online Pro and OpenShift Container Platform. This example is not currently available on OpenShift Online Starter.

Example proficiency level: **Foundational**.

The *Circuit Breaker* example 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 example shows you how to implement a Circuit Breaker and Fallback pattern in your services.

13.5.1. 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.

13.5.2. Circuit Breaker design tradeoffs

Table 13.4. 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.

13.5.3. Deploying the Circuit Breaker example application to OpenShift Online

Use one of the following options to execute the Circuit Breaker example application 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 deployment workflow that executes the **oc** commands for you.

13.5.3.1. Deploying the example application using developers.redhat.com/launch

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Red Hat Developer Launcher web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the developers.redhat.com/launch URL in a browser.

2. Follow on-screen instructions to create and launch your example application in Thorntail.

13.5.3.2. Authenticating the oc CLI client

To work with example applications on OpenShift Online using the **oc** command-line client, you must 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

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

13.5.3.3. Deploying the Circuit Breaker example application using the oc CLI client

This section shows you how to build your Circuit Breaker example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using developers.redhat.com/launch. For more information, see Section 13.5.3.1, “Deploying the example application using developers.redhat.com/launch” .
- The **oc** client authenticated. For more information, see Section 13.5.3.2, “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 application.

4. Use Maven to start the deployment to OpenShift.

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

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

5. Check the status of your application 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. After your example application 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.

13.5.4. Deploying the Circuit Breaker example application to Minishift or CDK

Use one of the following options to execute the Circuit Breaker example application locally on Minishift or CDK:

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

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

13.5.4.1. Getting the Launcher tool URL and credentials

You need the Launcher tool URL and user credentials to create and deploy example applications on Minishift or CDK. This information is provided when the Minishift or CDK is started.

Prerequisites

- The Launcher tool installed, configured, and running.

Procedure

1. Navigate to the console where you started Minishift or CDK.
2. Check the console output for the URL and user credentials you can use to access the running Launcher:

   Example Console Output from a Minishift or CDK 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
   ```

13.5.4.2. Deploying the example application using the Launcher tool

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Launcher web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.5.4.1, “Getting the Launcher tool URL and credentials” .

Procedure

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

13.5.4.3. Authenticating the oc CLI client

To work with example applications on Minishift or CDK using the **oc** command-line client, you must authenticate the client using the token provided by the Minishift or CDK web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.5.4.1, “Getting the Launcher tool URL and credentials” .

Procedure

1. Navigate to the Minishift or CDK 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your Minishift or CDK account.

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

13.5.4.4. Deploying the Circuit Breaker example application using the oc CLI client

This section shows you how to build your Circuit Breaker example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using Launcher tool on a Minishift or CDK. For more information, see Section 13.5.4.2, “Deploying the example application using the Launcher tool” .
- Your Launcher tool URL.
- The **oc** client authenticated. For more information, see Section 13.5.4.3, “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 application.
4. Use Maven to start the deployment to OpenShift.

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

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

5. Check the status of your application 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. After your example application 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.

13.5.5. Deploying the Circuit Breaker example application to OpenShift Container Platform

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

Prerequisites

- The example application created using developers.redhat.com/launch.

Procedure

- Follow the instructions in Section 13.5.3, “Deploying the Circuit Breaker example application to OpenShift Online”, only use the URL and user credentials from the OpenShift Container Platform Web Console.

13.5.6. Interacting with the unmodified Thorntail Circuit Breaker example application

After you have the Thorntail example application 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!"}
   ```

13.5.7. Running the Circuit Breaker example application integration tests

This example application 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 example application from the target OpenShift project. To avoid accidentally removing your example 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
```

13.5.8. 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 for Spring Boot
- Circuit Breaker for Eclipse Vert.x
- Circuit Breaker for Node.js

13.6. SECURED EXAMPLE APPLICATION FOR THORNTAIL

IMPORTANT

The following example is not meant to be run in a production environment.

Limitation: Run this example application on a Minishift or CDK. You can also use a manual workflow to deploy this example to OpenShift Online Pro and OpenShift Container Platform. This example is not currently available on OpenShift Online Starter.

Example proficiency level: **Advanced**.

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

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

13.6.1. The Secured project structure

The SSO example 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

13.6.2. Red Hat SSO deployment configuration

The **service.sso.yaml** file in this example 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 example security configuration impact the ability to use it in a production environment.

Table 13.5. SSO Example Simplifications

Change	Reason	Recommendation

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.
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 examples 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.

13.6.3. Red Hat SSO realm model

The **demo** realm is used to secure this example. 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 **demo** realm that can be used to validate various authentication and authorization outcomes: **admin** and **alice**.

13.6.3.1. Red Hat SSO users

The realm model for the secured examples 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/demo", 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
      "example-admin"
    ]
  },
  "resource_access": { 3
    "secured-example-endpoint": {
      "roles": [
        "example-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 **example-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-example-endpoint.roles** object contains the roles granted to **alice** for the **secured-example-endpoint** resource.

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

13.6.3.2. 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 **demo** 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. The token contains grants for the **alice** user which enable **alice** to access the Thorntail, Eclipse Vert.x, Node.js and Spring Boot based REST example application deployments.

secured-example-endpoint

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

13.6.4. 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>example-admin</role-name> 2
    </auth-constraint>
  </security-constraint>

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

  <security-role>
    <role-name>example-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": "demo", 1
  "resource": "secured-example-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.

13.6.5. Deploying the Secured example application to Minishift or CDK

13.6.5.1. Getting the Launcher tool URL and credentials

You need the Launcher tool URL and user credentials to create and deploy example applications on Minishift or CDK. This information is provided when the Minishift or CDK is started.

Prerequisites

- The Launcher tool installed, configured, and running.

Procedure

1. Navigate to the console where you started Minishift or CDK.

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

   Example Console Output from a Minishift or CDK 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
```

13.6.5.2. Creating the Secured example application using Launcher

Prerequisites

- The URL and user credentials of your running Launcher instance. For more information, see Section 13.6.5.1, "Getting the Launcher tool URL and credentials" .

Procedure

- Navigate to the Launcher URL in a browser.
- Follow the on-screen instructions to create your example 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.

13.6.5.3. Authenticating the oc CLI client

To work with example applications on Minishift or CDK using the **oc** command-line client, you must authenticate the client using the token provided by the Minishift or CDK web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.6.5.1, "Getting the Launcher tool URL and credentials" .

Procedure

1. Navigate to the Minishift or CDK 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your Minishift or CDK account.

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

13.6.5.4. Deploying the Secured example application using the oc CLI client

This section shows you how to build your Secured example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using the Launcher tool on a Minishift or CDK. For more information, see Section 13.6.5.2, "Creating the Secured example application using Launcher" .
- Your Launcher URL.
- The **oc** client authenticated. For more information, see Section 13.6.5.3, "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 application.
4. Deploy the Red Hat SSO server using the **service.sso.yaml** file from your example ZIP file:

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

5. Use Maven to start the deployment to Minishift or CDK.

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

   This command uses the OpenShift Maven plugin to launch the S2I process on Minishift or CDK 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 Minishift or CDK server.

13.6.6. Deploying the Secured example application to OpenShift Container Platform

In addition to the Minishift or CDK, you can create and deploy the example on OpenShift Container Platform with only minor differences. The most important difference is that you need to create the example application on Minishift or CDK before you can deploy it with OpenShift Container Platform.

Prerequisites

- The example created using Minishift or CDK.

13.6.6.1. Authenticating the oc CLI client

To work with example applications on OpenShift Container Platform using the **oc** command-line client, you must 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. Copy the **oc login** command.
5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Container Platform account.

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

13.6.6.2. Deploying the Secured example application using the oc CLI client

This section shows you how to build your Secured example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using the Launcher tool on a Minishift or CDK.
- The **oc** client authenticated. For more information, see Section 13.6.6.1, "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 application.

4. Deploy the Red Hat SSO server using the **service.sso.yaml** file from your example ZIP file:

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

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

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

   This command uses the OpenShift 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.

13.6.7. Authenticating to the Secured example application API endpoint

The Secured example application 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 example application using the access token returned by the authentication step.

13.6.7.1. Getting the Secured example application API endpoint

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

Prerequisites

- The Secured example application 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 13.1. 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 example endpoint would be **http://PROJECT_ID-myproject.LOCAL_OPENSHIFT_HOSTNAME**. **PROJECT_ID** is based on the name you entered when generating your example using developers.redhat.com/launch or the Launcher tool.

13.6.7.2. 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 example endpoint URL. For more information, see Section 13.6.7.1, “Getting the Secured example application 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/demo/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'

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRek1nbXhZMUhrQnpxTnR0SnkwMm5j
NTNtMGNiWDQxV1hNSTU1MFo4MGVBIn0.eyJqdGkiOiI0NDA3YTliNC04YWRhLTRlMTctOD
Q2ZS03YjI5MjMyN2RmYTIiLCJleHAiOjE1MDc3OTM3ODcsIm5iZiI6MCwiaWF0IjoxNTA3Nzkz
```

```
NzI3LCJpc3MiOiJodHRwczovL3NlY3VyZS1zc28tc3NvLWRlbW8uYXBwcy5jYWZlLWJhYmUub
3JnL2F1dGgvcmVhbG1zL21hc3RlciIsImF1ZCI6ImRlbW9hcHAiLCJzdWIiOiJjMDE3NWNjYi0w
ODkyLTRiMzEtODI5Zi1kZGE4NzM4MTVmZTgiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJkZW1vY
XBwIiwiYXV0aF90aW1lIjowLCJzZXNzaW9uX3N0YXRlIjoiMDFjOTkzNGQtNmZmOS00NWYzL
WJkNWUtMTU4NDI5ZDZjNDczIiwiYWNyIjoiMSIsImNsaWVudF9zZXNzaW9uIjoiMzM3Yzk0MT
YtYTdlZS00ZWUzLThjZWQtODhlODI0MGJjNTAyIiwiYWxsb3dlZC1vcmlnaW5zIjpbIioiXSwicmV
hbG1fYWNjZXNzIjp7InJvbGVzIjpbImJvb3N0ZXItYWRtaW4iXX0sInJlc291cmNlX2FjY2VzcyI6ey
JzZWN1cmVkLWJvb3N0ZXItZW5kcG9pbnQiOnsicm9sZXMiOlsiYm9vc3Rlci1hZG1pbiJdfSwiY
WNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsInZpZXctcHJvZmlsZSJdfX0sIm5hbW
UiOiJBbGljZSBJbkNoYWlucyIsInByZWZlcnJlZF91c2VybmFtZSI6ImFsaWNlIiwiZ2l2ZW5fbmFtZ
SI6IkFsaWNlIiwiZmFtaWx5X25hbWUiOiJJbkNoYWlucyIsImVtYWlsIjoiYWxpY2VAa2V5Y2xvYW
sub3JnIn0.mjmZe37enHpigJv0BGuIitOj-
kfMLPNwYzNd3n0Ax4Nga7KpnfytGyuPSvR4KAG8rzkfBNN9kIPYdy7pJEeYlfmnFUkM4EDrZY
gn4qZAznP1Wzy1RfVRdUFi0-
GqFTMPb37o5HRldZZ09QljX_j3GHnoMGXRtYW9RZN4eKkYkcz9hRwgfJoTy2CuwFqeJwZY
UyXifrfA-JoTr0UmSUed-0NMksGrtJjjPggUGS-
qOn6OgKcmN2vaVAQlxW32y53JqUXctfLQ6DhJzIMYTmOflIPy0sgG1mG7sovQhw1xTg0vTjdx
8zQ-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 example 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 13.2. 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 example endpoint. For more information, see Section 13.6.7.1, “Getting the Secured example application 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/cs30OzJ
9olkteLVNRjzdTxFOyRtS2ovDfzdhhO9XzUcTMbIsCOAZtSt8K+6yvBXypOSYvI75EUdypm
kcK1KoptqY5KEBQ1KwhWuP7IWQ0fshUwD6jI1QWDfGxfM/h34FvEn/0tJ71xN2P8TI2Yan
wuDZgosdobx/PAvlGREBGuk4BgmexTOkAdnFxIUQcCkiEZ2C41uCrxiS4CEe5OX91aK9
HKZV4ZJX6vnqMHmdDnsMdO+UFtxOBYZio+a1jP4W3d7J5fGeiOaXjQCOpivKnP2yU2D
PdWmDMyVb67l8DRA+jh0OJFKZ5H2fNgE3II59vdsRwIDAQAB
-----END PUBLIC KEY-----
```

   > NOTE
   >
   > This is the **demo** realm public key from the Red Hat SSO server deployment of the Secured example application.

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

13.6.7.3. 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 13.6.7.1, “Getting the Secured example application API endpoint”.

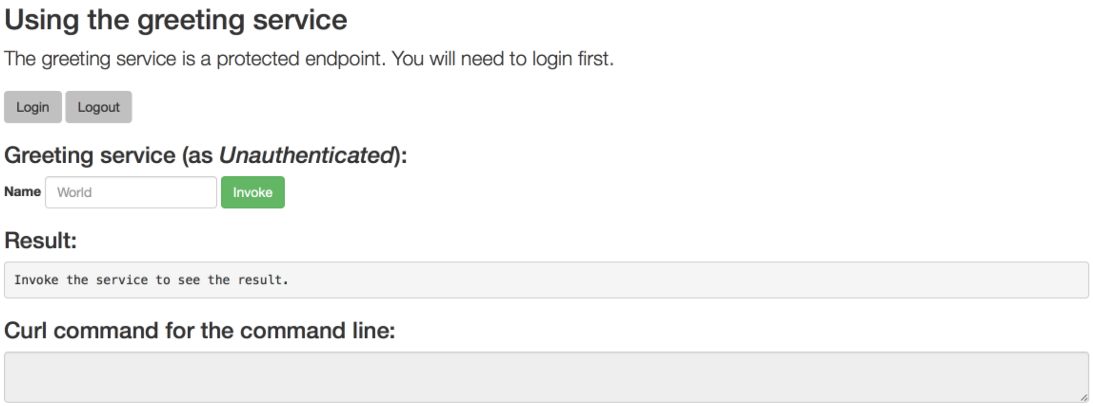
Procedure

1. In a web browser, navigate to the endpoint URL.

2. Perform an unauthenticated request:

   a. Click the *Invoke* button.

Figure 13.1. Unauthenticated Secured Example Web Interface



Using the greeting service

The greeting service is a protected endpoint. You will need to login first.

Greeting service (as *Unauthenticated*):

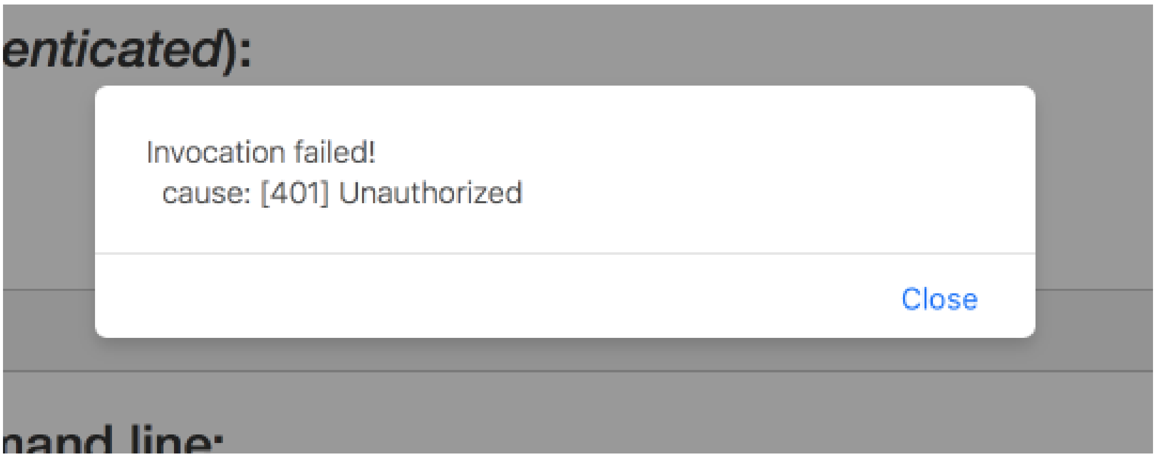
Result:

```
Invoke the service to see the result.
```

Curl command for the command line:

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

Figure 13.2. Unauthenticated Error Message



Invocation failed!
cause: [401] Unauthorized

3. Perform an authenticated request as a user:

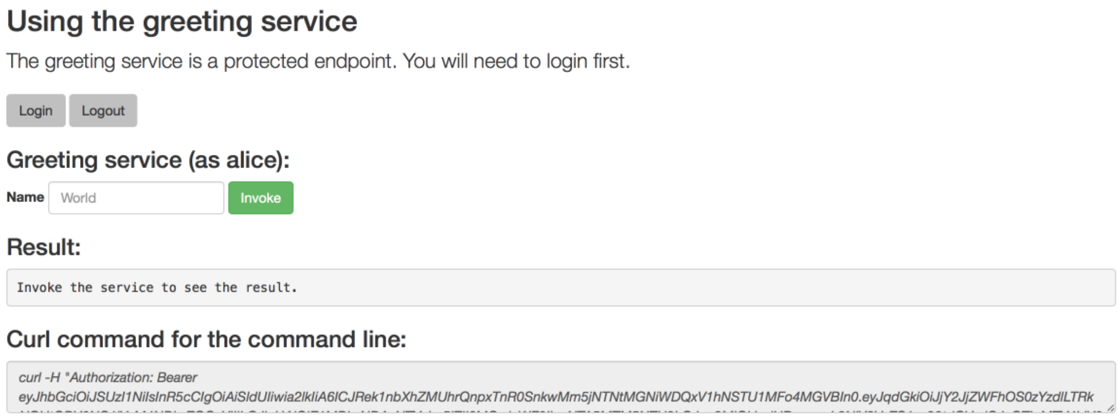
   a. Click the *Login* button to authenticate against Red Hat SSO. You will be redirected to the SSO server.

   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 13.3. Authenticated Secured Example Web Interface (as Alice)


Using the greeting service

The greeting service is a protected endpoint. You will need to login first.

Greeting service (as alice):

Result:

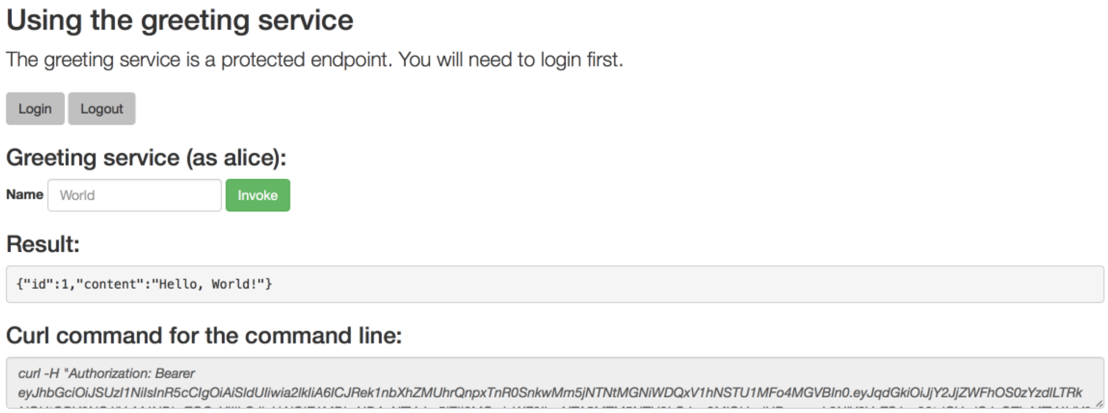
```
Invoke the service to see the result.
```

Curl command for the command line:

```
curl -H "Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRek1nbXhZMUhrQnpxTnR0SnkwMm5jNTNtMGNiWDQxV1hNSTU1MFo4MGVBIn0.eyJqdGkiOiJjY2JjZWFhOS0zYzdlLTRk
```

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


Using the greeting service

The greeting service is a protected endpoint. You will need to login first.

Greeting service (as alice):

Result:

```
{"id":1,"content":"Hello, World!"}
```

Curl command for the command line:

```
curl -H "Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRek1nbXhZMUhrQnpxTnR0SnkwMm5jNTNtMGNiWDQxV1hNSTU1MFo4MGVBIn0.eyJqdGkiOiJjY2JjZWFhOS0zYzdlLTRk
```

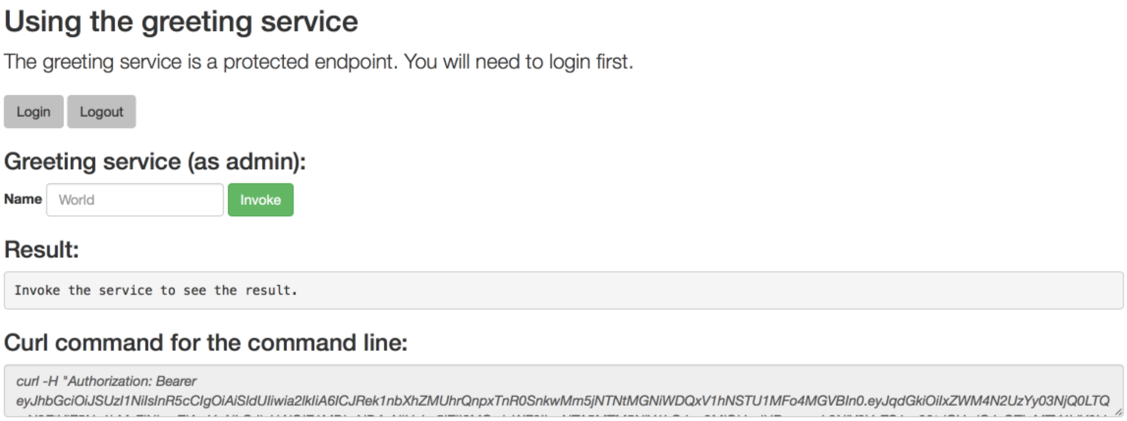
d. Log out.

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 13.5. Authenticated Secured Example Web Interface (as admin)



Using the greeting service

The greeting service is a protected endpoint. You will need to login first.

Greeting service (as admin):

Name: World

Result:

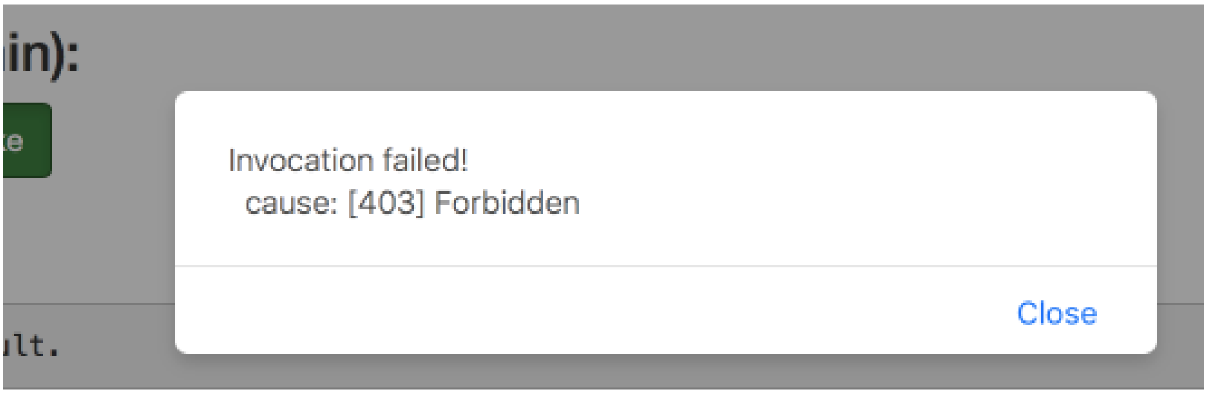
```
Invoke the service to see the result.
```

Curl command for the command line:

```
curl -H "Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRek1nbXhZMUhrQnpxTnR0SnkwMm5jNTNtMGNiWDQxV1hNSTU1MFo4MGVBIn0.eyJqdGkiOiIxZWM4N2UzYy03NjQ0LTQ
```

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 13.6. Unauthorized Error Message

   Invocation failed!
   cause: [403] Forbidden

13.6.8. Running the Thorntail Secured example application integration tests

IMPORTANT

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

This section shows you how to execute the integration tests using Red Hat SSO server provided by the authenticated oc client.

Prerequisites

- The **oc** client authenticated.

PROCEDURE

Executing integration tests removes all existing instances of the example application from the target OpenShift project. To avoid accidentally removing your example 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
   ```

13.6.9. 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 for Spring Boot
- Secured for Eclipse Vert.x
- Secured for Node.js

13.7. CACHE EXAMPLE FOR THORNTAIL

IMPORTANT

The following example is not meant to be run in a production environment.

Limitation: Run this example application on a Minishift or CDK. You can also use a manual workflow to deploy this example to OpenShift Online Pro and OpenShift Container Platform. This example is not currently available on OpenShift Online Starter.

Example proficiency level: **Advanced**.

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

This example shows you how to:

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

13.7.1. 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 13.3. 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 example 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.

13.7.2. Deploying the Cache example application to OpenShift Online

Use one of the following options to execute the Cache example application 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 deployment workflow that executes the **oc** commands for you.

13.7.2.1. Deploying the example application using developers.redhat.com/launch

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Red Hat Developer Launcher web interface.

Prerequisites

- An account at OpenShift Online.

Procedure

1. Navigate to the developers.redhat.com/launch URL in a browser.

2. Follow on-screen instructions to create and launch your example application in Thorntail.

13.7.2.2. Authenticating the oc CLI client

To work with example applications on OpenShift Online using the **oc** command-line client, you must 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. Copy the **oc login** command.

5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your OpenShift Online account.

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

13.7.2.3. Deploying the Cache example application using the oc CLI client

This section shows you how to build your Cache example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using developers.redhat.com/launch. For more information, see Section 13.7.2.1, "Deploying the example application using developers.redhat.com/launch" .

- The **oc** client authenticated. For more information, see Section 13.7.2.2, "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 application.

4. Deploy the cache service.

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

NOTE

If you are using an architecture other than x86_64, in the YAML file, update the image name of Red Hat Data Grid to its relevant image name in that architecture. For example, for the s390x and ppc64le architecture, update the image name to its IBM Z and IBM Power Systems image name **registry.access.redhat.com/jboss-datagrid-7/datagrid73-openj9-11-openshift-rhel8**.

5. Use Maven to start the deployment to OpenShift.

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

6. Check the status of your application 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. After your example application 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.

13.7.3. Deploying the Cache example application to Minishift or CDK

Use one of the following options to execute the Cache example application locally on Minishift or CDK:

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

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

13.7.3.1. Getting the Launcher tool URL and credentials

You need the Launcher tool URL and user credentials to create and deploy example applications on Minishift or CDK. This information is provided when the Minishift or CDK is started.

Prerequisites

- The Launcher tool installed, configured, and running.

Procedure

1. Navigate to the console where you started Minishift or CDK.

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

Example Console Output from a Minishift or CDK 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
```

13.7.3.2. Deploying the example application using the Launcher tool

This section shows you how to build your REST API Level 0 example application and deploy it to OpenShift from the Launcher web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.7.3.1, “Getting the Launcher tool URL and credentials” .

Procedure

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

13.7.3.3. Authenticating the oc CLI client

To work with example applications on Minishift or CDK using the **oc** command-line client, you must authenticate the client using the token provided by the Minishift or CDK web interface.

Prerequisites

- The URL of your running Launcher instance and the user credentials of your Minishift or CDK. For more information, see Section 13.7.3.1, “Getting the Launcher tool URL and credentials” .

Procedure

1. Navigate to the Minishift or CDK 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. Copy the **oc login** command.

5. Paste the command in a terminal. The command uses your authentication token to authenticate your **oc** CLI client with your Minishift or CDK account.

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

13.7.3.4. Deploying the Cache example application using the oc CLI client

This section shows you how to build your Cache example application and deploy it to OpenShift from the command line.

Prerequisites

- The example application created using Launcher tool on a Minishift or CDK. For more information, see Section 13.7.3.2, “Deploying the example application using the Launcher tool” .

- Your Launcher tool URL.

- The **oc** client authenticated. For more information, see Section 13.7.3.3, “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 application.

4. Deploy the cache service.

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

> **NOTE**
>
> If you are using an architecture other than x86_64, in the YAML file, update the image name of Red Hat Data Grid to its relevant image name in that architecture. For example, for the s390x and ppc64le architecture, update the image name to its IBM Z and IBM Power Systems image name **registry.access.redhat.com/jboss-datagrid-7/datagrid73-openj9-11-openshift-rhel8**.

5. Use Maven to start the deployment to OpenShift.

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

6. Check the status of your application 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. After your example application 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.

13.7.4. Deploying the Cache example application to OpenShift Container Platform

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

Prerequisites

- The example application created using developers.redhat.com/launch.

Procedure

- Follow the instructions in Section 13.7.2, "Deploying the Cache example application to OpenShift Online", only use the URL and user credentials from the OpenShift Container Platform Web Console.

13.7.5. Interacting with the unmodified Cache example application

Use the default web interface to interact with the unmodified Cache example application, and see how storing frequently accessed data can shorten the time needed to access your service.

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 from **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.

13.7.6. Running the Cache example application integration tests

This example application 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 example application from the target OpenShift project. To avoid accidentally removing your example 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
```

13.7.7. Caching resources

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

- Cache for Spring Boot
- Cache for Eclipse Vert.x
- Cache for Node.js

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 AN EXAMPLE APPLICATION

The deployment configuration for an example application contains information related to deploying and running the application in OpenShift, such as route information or readiness probe location. The deployment configuration of an example application is stored in a set of YAML files. For examples that use the OpenShift Maven plugin, the YAML files are located in the **src/main/jkube/** directory. For examples using Nodeshift, the YAML files are located in the **.nodeshift** directory.

IMPORTANT

The deployment configuration files used by the OpenShift Maven plugin and Nodeshift do not have to be full OpenShift resource definitions. Both OpenShift 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 OpenShift Maven plugin are available in the **target/classes/META-INF/jkube/** directory. The resource definitions generated by Nodeshift are available in the **tmp/nodeshift/resource/** directory.

Prerequisites

- An existing example 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 example 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 example using Maven or npm.
3. Verify that your configuration updates show in the deployed version of your example.

   ```
   $ oc export all --as-template='my-template'

   apiVersion: template.openshift.io/v1
   kind: Template
   ```

```
metadata:
  creationTimestamp: null
  name: my-template
objects:
- apiVersion: template.openshift.io/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 OPENSHIFT MAVEN PLUGIN

Similar to using Maven and the OpenShift Maven plugin from your local host to deploy an application, you can configure Jenkins to use Maven and the OpenShift 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 OpenShift Maven plugin in the **pom.xml**, and built using a RHEL base image.

  NOTE

  For building and deploying your applications to OpenShift, Thorntail 2.7 only supports builder images based on OpenJDK 8 and OpenJDK 11. Oracle JDK and OpenJDK 9 builder images are not supported.

  Example pom.xml

  ```
  <properties>
    ...
    <jkube.generator.from>registry.access.redhat.com/redhat-openjdk-18/openjdk18-openshift:latest</jkube.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 oc:deploy -Popenshift -Djkube.namespace=MY_PROJECT
```

Substitute **MY_PROJECT** with the name of the OpenShift project for your application.

a. 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 Chapter 8, *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

thorntail.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.

thorntail.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.

thorntail.cdi.require-bean-descriptor

If true then implicit bean archives without bean descriptor file (beans.xml) are ignored by Weld

thorntail.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:

```
thorntail:
  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:

```
thorntail:
  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.PostgreSQLValidConnectionChecker
        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:

```
thorntail:
  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

thorntail.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

thorntail.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

thorntail.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

thorntail.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.

thorntail.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.

thorntail.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

thorntail.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

thorntail.datasources.data-sources.*KEY*.capacity-decrementer-class

Class defining the policy for decrementing connections in the pool

thorntail.datasources.data-sources.*KEY*.capacity-decrementer-properties

Properties to be injected in class defining the policy for decrementing connections in the pool

thorntail.datasources.data-sources.*KEY*.capacity-incrementer-class

Class defining the policy for incrementing connections in the pool

thorntail.datasources.data-sources.*KEY*.capacity-incrementer-properties

Properties to be injected in class defining the policy for incrementing connections in the pool

thorntail.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

thorntail.datasources.data-sources.*KEY*.connectable

Enable the use of CMR. This feature means that a local resource can reliably participate in an XA transaction.

thorntail.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.

thorntail.datasources.data-sources.*KEY*.connection-listener-property

Properties to be injected in class specidied in connection-listener-class

thorntail.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

thorntail.datasources.data-sources.*KEY*.connection-url

The JDBC driver connection URL

thorntail.datasources.data-sources.*KEY*.credential-reference

Credential (from Credential Store) to authenticate on data source

thorntail.datasources.data-sources.*KEY*.datasource-class

The fully qualified name of the JDBC datasource class

thorntail.datasources.data-sources.*KEY*.driver-class

The fully qualified name of the JDBC driver class

thorntail.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

thorntail.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).

thorntail.datasources.data-sources.*KEY*.enabled

Specifies if the datasource should be enabled. Note this attribute will not be supported runtime in next versions.

thorntail.datasources.data-sources.*KEY*.enlistment-trace

Defines if WildFly/IronJacamar should record enlistment traces

thorntail.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

thorntail.datasources.data-sources.*KEY*.exception-sorter-properties

The exception sorter properties

thorntail.datasources.data-sources.*KEY*.flush-strategy

Specifies how the pool should be flush in case of an error.

thorntail.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.

thorntail.datasources.data-sources.*KEY*.initial-pool-size

The initial-pool-size element indicates the initial number of connections a pool should hold.

thorntail.datasources.data-sources.*KEY*.jndi-name

Specifies the JNDI name for the datasource

thorntail.datasources.data-sources.*KEY*.jta

Enable JTA integration

thorntail.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

thorntail.datasources.data-sources.*KEY*.mcp

Defines the ManagedConnectionPool implementation, f.ex. org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool

thorntail.datasources.data-sources.*KEY*.min-pool-size

The min-pool-size element specifies the minimum number of connections for a pool

thorntail.datasources.data-sources.*KEY*.new-connection-sql

Specifies an SQL statement to execute whenever a connection is added to the connection pool

thorntail.datasources.data-sources.*KEY*.password

Specifies the password used when creating a new connection

thorntail.datasources.data-sources.*KEY*.pool-fair

Defines if pool use should be fair

thorntail.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.

thorntail.datasources.data-sources.*KEY*.pool-use-strict-min

Specifies if the min-pool-size should be considered strictly

thorntail.datasources.data-sources.*KEY*.prepared-statements-cache-size

The number of prepared statements per connection in an LRU cache

thorntail.datasources.data-sources.*KEY*.query-timeout

Any configured query timeout in seconds. If not provided no timeout will be set

thorntail.datasources.data-sources.*KEY*.reauth-plugin-class-name

The fully qualified class name of the reauthentication plugin implementation

thorntail.datasources.data-sources.*KEY*.reauth-plugin-properties

The properties for the reauthentication plugin

thorntail.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

thorntail.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

thorntail.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

thorntail.datasources.data-sources.*KEY*.spy

Enable spying of SQL statements

thorntail.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

thorntail.datasources.data-sources.*KEY*.stale-connection-checker-properties

The stale connection checker properties

thorntail.datasources.data-sources.*KEY*.statistics-enabled

Define whether runtime statistics are enabled or not.

thorntail.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

thorntail.datasources.data-sources.*KEY*.tracking

Defines if IronJacamar should track connection handles across transaction boundaries

thorntail.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

thorntail.datasources.data-sources.*KEY*.url-delimiter

Specifies the delimiter for URLs in connection-url for HA datasources

thorntail.datasources.data-sources.*KEY*.url-selector-strategy-class-name

A class that implements org.jboss.jca.adapters.jdbc.URLSelectorStrategy

thorntail.datasources.data-sources.*KEY*.use-ccm

Enable the use of a cached connection manager

thorntail.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)

thorntail.datasources.data-sources.*KEY*.use-java-context

Setting this to false will bind the datasource into global JNDI

thorntail.datasources.data-sources.*KEY*.use-try-lock

Any configured timeout for internal locks on the resource adapter objects in seconds

thorntail.datasources.data-sources.*KEY*.user-name

Specify the user name used when creating a new connection

thorntail.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

thorntail.datasources.data-sources.*KEY*.valid-connection-checker-properties

The valid connection checker properties

thorntail.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

thorntail.datasources.installed-drivers

List of JDBC drivers that have been installed in the runtime

thorntail.datasources.jdbc-drivers.*KEY*.datasource-class-info

The available properties for the datasource-class, and xa-datasource-class for the jdbc-driver

thorntail.datasources.jdbc-drivers.*KEY*.deployment-name

The name of the deployment unit from which the driver was loaded

thorntail.datasources.jdbc-drivers.*KEY*.driver-class-name

The fully qualified class name of the java.sql.Driver implementation

thorntail.datasources.jdbc-drivers.*KEY*.driver-datasource-class-name

The fully qualified class name of the javax.sql.DataSource implementation

thorntail.datasources.jdbc-drivers.*KEY*.driver-major-version

The driver’s major version number

thorntail.datasources.jdbc-drivers.*KEY*.driver-minor-version

The driver’s minor version number

thorntail.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

thorntail.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

thorntail.datasources.jdbc-drivers.*KEY*.driver-xa-datasource-class-name

The fully qualified class name of the javax.sql.XADataSource implementation

thorntail.datasources.jdbc-drivers.*KEY*.jdbc-compliant

Whether or not the driver is JDBC compliant

thorntail.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

thorntail.datasources.jdbc-drivers.*KEY*.profile

Domain Profile in which driver is defined. Null in case of standalone server

thorntail.datasources.jdbc-drivers.*KEY*.xa-datasource-class

XA datasource class

thorntail.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

thorntail.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

thorntail.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

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.capacity-decrementer-class

Class defining the policy for decrementing connections in the pool

thorntail.datasources.xa-data-sources.*KEY*.capacity-decrementer-properties

Properties to inject in class defining the policy for decrementing connections in the pool

thorntail.datasources.xa-data-sources.*KEY*.capacity-incrementer-class

Class defining the policy for incrementing connections in the pool

thorntail.datasources.xa-data-sources.*KEY*.capacity-incrementer-properties

Properties to inject in class defining the policy for incrementing connections in the pool

thorntail.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

thorntail.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.

thorntail.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.

thorntail.datasources.xa-data-sources.*KEY*.connection-listener-property

Properties to be injected in class specified in connection-listener-class

thorntail.datasources.xa-data-sources.*KEY*.credential-reference

Credential (from Credential Store) to authenticate on data source

thorntail.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

thorntail.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).

thorntail.datasources.xa-data-sources.*KEY*.enabled

Specifies if the datasource should be enabled. Note this attribute will not be supported runtime in next versions.

thorntail.datasources.xa-data-sources.*KEY*.enlistment-trace

Defines if WildFly/IronJacamar should record enlistment traces

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.exception-sorter-properties

The exception sorter properties

thorntail.datasources.xa-data-sources.*KEY*.flush-strategy

Specifies how the pool should be flush in case of an error.

thorntail.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.

thorntail.datasources.xa-data-sources.*KEY*.initial-pool-size

The initial-pool-size element indicates the initial number of connections a pool should hold.

thorntail.datasources.xa-data-sources.*KEY*.interleaving

An element to enable interleaving for XA connections

thorntail.datasources.xa-data-sources.*KEY*.jndi-name

Specifies the JNDI name for the datasource

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.mcp

Defines the ManagedConnectionPool implementation, f.ex. org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool

thorntail.datasources.xa-data-sources.*KEY*.min-pool-size

The min-pool-size element specifies the minimum number of connections for a pool

thorntail.datasources.xa-data-sources.*KEY*.new-connection-sql

Specifies an SQL statement to execute whenever a connection is added to the connection pool

thorntail.datasources.xa-data-sources.*KEY*.no-recovery

Specifies if the connection pool should be excluded from recovery

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.pad-xid

Should the Xid be padded

thorntail.datasources.xa-data-sources.*KEY*.password

Specifies the password used when creating a new connection

thorntail.datasources.xa-data-sources.*KEY*.pool-fair

Defines if pool use should be fair

thorntail.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.

thorntail.datasources.xa-data-sources.*KEY*.pool-use-strict-min

Specifies if the min-pool-size should be considered strictly

thorntail.datasources.xa-data-sources.*KEY*.prepared-statements-cache-size

The number of prepared statements per connection in an LRU cache

thorntail.datasources.xa-data-sources.*KEY*.query-timeout

Any configured query timeout in seconds. If not provided no timeout will be set

thorntail.datasources.xa-data-sources.*KEY*.reauth-plugin-class-name

The fully qualified class name of the reauthentication plugin implementation

thorntail.datasources.xa-data-sources.*KEY*.reauth-plugin-properties

The properties for the reauthentication plugin

thorntail.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.

thorntail.datasources.xa-data-sources.*KEY*.recovery-credential-reference

Credential (from Credential Store) to authenticate on data source

thorntail.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).

thorntail.datasources.xa-data-sources.*KEY*.recovery-password

The password used for recovery

thorntail.datasources.xa-data-sources.*KEY*.recovery-plugin-class-name

The fully qualified class name of the recovery plugin implementation

thorntail.datasources.xa-data-sources.*KEY*.recovery-plugin-properties

The properties for the recovery plugin

thorntail.datasources.xa-data-sources.*KEY*.recovery-security-domain

The security domain used for recovery

thorntail.datasources.xa-data-sources.*KEY*.recovery-username

The user name used for recovery

thorntail.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

thorntail.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

thorntail.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

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.spy

Enable spying of SQL statements

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.stale-connection-checker-properties

The stale connection checker properties

thorntail.datasources.xa-data-sources.*KEY*.statistics-enabled

Define whether runtime statistics are enabled or not.

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.tracking

Defines if IronJacamar should track connection handles across transaction boundaries

thorntail.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.

thorntail.datasources.xa-data-sources.*KEY*.url-delimiter

Specifies the delimiter for URLs in connection-url for HA datasources

thorntail.datasources.xa-data-sources.*KEY*.url-property

Specifies the property for the URL property in the xa-datasource-property values

thorntail.datasources.xa-data-sources.*KEY*.url-selector-strategy-class-name

A class that implements org.jboss.jca.adapters.jdbc.URLSelectorStrategy

thorntail.datasources.xa-data-sources.*KEY*.use-ccm

Enable the use of a cached connection manager

thorntail.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)

thorntail.datasources.xa-data-sources.*KEY*.use-java-context

Setting this to false will bind the datasource into global JNDI

thorntail.datasources.xa-data-sources.*KEY*.use-try-lock

Any configured timeout for internal locks on the resource adapter objects in seconds

thorntail.datasources.xa-data-sources.*KEY*.user-name

Specify the user name used when creating a new connection

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.valid-connection-checker-properties

The valid connection checker properties

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.wrap-xa-resource

Should the XAResource instances be wrapped in an org.jboss.tm.XAResourceWrapper instance

thorntail.datasources.xa-data-sources.*KEY*.xa-datasource-class

The fully qualified name of the javax.sql.XADataSource implementation

thorntail.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

thorntail.datasources.xa-data-sources.*KEY*.xa-resource-timeout

The value is passed to XAResource.setTransactionTimeout(), in seconds. Default is zero

thorntail.ds.connection.url

Default datasource connection URL

thorntail.ds.name

Name of the default datasource

thorntail.ds.password

Default datasource connection password

thorntail.ds.username

Default datasource connection user name

thorntail.jdbc.driver

Default datasource JDBC driver name

D.7. DISCOVERY

An internal fraction for WildFly Discovery.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>discovery</artifactId>
</dependency>
```

Configuration

thorntail.discovery.aggregate-providers.*KEY*.providers

The list of providers that are aggregated by this provider

thorntail.discovery.static-providers.*KEY*.services

The list of services that are configured with this provider

D.8. 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

thorntail.ee.annotation-property-replacement

Flag indicating whether Jakarta EE annotations will have property replacements applied

thorntail.ee.context-services.*KEY*.jndi-name

The JNDI Name to lookup the context service.

thorntail.ee.context-services.*KEY*.use-transaction-setup-provider

Flag which indicates if the transaction setup provider should be used

thorntail.ee.default-bindings-service.context-service

The JNDI name where the default EE Context Service can be found

thorntail.ee.default-bindings-service.datasource

The JNDI name where the default EE Datasource can be found

thorntail.ee.default-bindings-service.jms-connection-factory

The JNDI name where the default EE JMS Connection Factory can be found

thorntail.ee.default-bindings-service.managed-executor-service

The JNDI name where the default EE Managed Executor Service can be found

thorntail.ee.default-bindings-service.managed-scheduled-executor-service

The JNDI name where the default EE Managed Scheduled Executor Service can be found

thorntail.ee.default-bindings-service.managed-thread-factory

The JNDI name where the default EE Managed Thread Factory can be found

thorntail.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.

thorntail.ee.global-modules

A list of modules that should be made available to all deployments.

thorntail.ee.jboss-descriptor-property-replacement

Flag indicating whether JBoss specific deployment descriptors will have property replacements applied

thorntail.ee.managed-executor-services.*KEY*.context-service

The name of the context service to be used by the executor.

thorntail.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.

thorntail.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.

thorntail.ee.managed-executor-services.*KEY*.jndi-name

The JNDI Name to lookup the managed executor service.

thorntail.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.

thorntail.ee.managed-executor-services.*KEY*.long-running-tasks

Flag which hints the duration of tasks executed by the executor.

thorntail.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).

thorntail.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.

thorntail.ee.managed-executor-services.*KEY*.reject-policy

The policy to be applied to aborted tasks.

thorntail.ee.managed-executor-services.*KEY*.thread-factory

The name of the thread factory to be used by the executor.

thorntail.ee.managed-scheduled-executor-services.*KEY*.context-service

The name of the context service to be used by the scheduled executor.

thorntail.ee.managed-scheduled-executor-services.*KEY*.core-threads

The number of threads to be used by the scheduled executor, even if they are idle. If this is not defined or is set to 0, the core pool size will be calculated based on the number of available processors.

thorntail.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.

thorntail.ee.managed-scheduled-executor-services.*KEY*.jndi-name

The JNDI Name to lookup the managed scheduled executor service.

thorntail.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.

thorntail.ee.managed-scheduled-executor-services.*KEY*.long-running-tasks

Flag which hints the duration of tasks executed by the scheduled executor.

thorntail.ee.managed-scheduled-executor-services.*KEY*.reject-policy

The policy to be applied to aborted tasks.

thorntail.ee.managed-scheduled-executor-services.*KEY*.thread-factory

The name of the thread factory to be used by the scheduled executor.

thorntail.ee.managed-thread-factories.*KEY*.context-service

The name of the context service to be used by the managed thread factory

thorntail.ee.managed-thread-factories.*KEY*.jndi-name

The JNDI Name to lookup the managed thread factory.

thorntail.ee.managed-thread-factories.*KEY*.priority

The priority applied to threads created by the factory

thorntail.ee.spec-descriptor-property-replacement

Flag indicating whether descriptors defined by the Jakarta EE specification will have property replacements applied

D.8.1. EE Security

Provides Java EE Security API support according to JSR 375.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>ee-security</artifactId>
</dependency>
```

D.9. EJB

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>ejb</artifactId>
</dependency>
```

Configuration

thorntail.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

thorntail.ejb3.application-security-domains.*KEY*.enable-jacc

Enable authorization using JACC

thorntail.ejb3.application-security-domains.*KEY*.referencing-deployments

The deployments currently referencing this mapping

thorntail.ejb3.application-security-domains.*KEY*.security-domain

The Elytron security domain to be used by deployments that reference the mapped security domain

thorntail.ejb3.async-service.thread-pool-name

The name of the thread pool which handles asynchronous invocations

thorntail.ejb3.caches.*KEY*.aliases

The aliases by which this cache may also be referenced

thorntail.ejb3.caches.*KEY*.passivation-store

The passivation store used by this cache

thorntail.ejb3.client-interceptors

Client interceptor definitions.

thorntail.ejb3.cluster-passivation-stores.*KEY*.bean-cache

The name of the cache used to store bean instances.

thorntail.ejb3.cluster-passivation-stores.*KEY*.cache-container

The name of the cache container used for the bean and client-mappings caches

thorntail.ejb3.cluster-passivation-stores.*KEY*.client-mappings-cache

The name of the cache used to store client-mappings of the EJB remoting connector's socket-bindings

thorntail.ejb3.cluster-passivation-stores.*KEY*.idle-timeout

The timeout in units specified by idle-timeout-unit, after which a bean will passivate

thorntail.ejb3.cluster-passivation-stores.*KEY*.idle-timeout-unit

The unit of idle-timeout

thorntail.ejb3.cluster-passivation-stores.*KEY*.max-size

The maximum number of beans this cache should store before forcing old beans to passivate

thorntail.ejb3.cluster-passivation-stores.*KEY*.passivate-events-on-replicate

Indicates whether replication should trigger passivation events on the bean

thorntail.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

thorntail.ejb3.default-distinct-name

The default distinct name that is applied to every EJB deployed on this server

thorntail.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

thorntail.ejb3.default-entity-bean-optimistic-locking

If set to true entity beans will use optimistic locking by default

thorntail.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

thorntail.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

thorntail.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

thorntail.ejb3.default-security-domain

The default security domain that will be used for EJBs if the bean doesn't explicitly specify one

thorntail.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

thorntail.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.

thorntail.ejb3.default-singleton-bean-access-timeout

The default access timeout for singleton beans

thorntail.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

thorntail.ejb3.default-stateful-bean-access-timeout

The default access timeout for stateful beans

thorntail.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

thorntail.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.

thorntail.ejb3.enable-statistics

If set to true, enable the collection of invocation statistics. Deprecated in favour of "statistics-enabled"

thorntail.ejb3.file-passivation-stores.*KEY*.groups-path

The subdirectory within the path specified by relative-to in which to store passivated bean groups

thorntail.ejb3.file-passivation-stores.*KEY*.idle-timeout

The timeout in units specified by idle-timeout-unit, after which a bean will passivate

thorntail.ejb3.file-passivation-stores.*KEY*.idle-timeout-unit

The unit of idle-timeout

thorntail.ejb3.file-passivation-stores.*KEY*.max-size

The maximum number of beans this cache should store before forcing old beans to passivate

thorntail.ejb3.file-passivation-stores.*KEY*.relative-to

The root path used to store passivated beans

thorntail.ejb3.file-passivation-stores.*KEY*.sessions-path

The subdirectory within the path specified by relative-to in which to store passivated beans

thorntail.ejb3.file-passivation-stores.*KEY*.subdirectory-count

Specifies the number of subdirectories into which stored state should be divided, used to minimize the number of files created per directory

thorntail.ejb3.identity-service.outflow-security-domains

References to security domains to attempt to outflow any established identity to

thorntail.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

thorntail.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)

thorntail.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.

thorntail.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)

thorntail.ejb3.mdb-delivery-groups.*KEY*.active

Indicates if delivery for all MDBs belonging to this group is active

thorntail.ejb3.passivation-stores.*KEY*.bean-cache

The name of the cache used to store bean instances.

thorntail.ejb3.passivation-stores.*KEY*.cache-container

The name of the cache container used for the bean and client-mappings caches

thorntail.ejb3.passivation-stores.*KEY*.max-size

The maximum number of beans this cache should store before forcing old beans to passivate

thorntail.ejb3.remote-service.channel-creation-options.*KEY*.type

The type of the channel creation option

thorntail.ejb3.remote-service.channel-creation-options.*KEY*.value

The value for the EJB remote channel creation option

thorntail.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

thorntail.ejb3.remote-service.connector-ref

The name of the connector on which the EJB3 remoting channel is registered

thorntail.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

thorntail.ejb3.remote-service.thread-pool-name

The name of the thread pool that handles remote invocations

thorntail.ejb3.remoting-profiles.*KEY*.exclude-local-receiver

If set no local receiver is used in this profile

thorntail.ejb3.remoting-profiles.*KEY*.local-receiver-pass-by-value

If set local receiver will pass ejb beans by value

thorntail.ejb3.remoting-profiles.*KEY*.remoting-ejb-receivers.*KEY*.channel-creation-options.*KEY*.type

The type of the channel creation option

thorntail.ejb3.remoting-profiles.*KEY*.remoting-ejb-receivers.*KEY*.channel-creation-options.*KEY*.value

The value for the EJB remote channel creation option

thorntail.ejb3.remoting-profiles.*KEY*.remoting-ejb-receivers.*KEY*.connect-timeout

Remoting ejb receiver connect timeout

thorntail.ejb3.remoting-profiles.*KEY*.remoting-ejb-receivers.*KEY*.outbound-connection-ref

Name of outbound connection that will be used by the ejb receiver

thorntail.ejb3.remoting-profiles.*KEY*.static-ejb-discovery

Describes static discovery config for EJB’s

thorntail.ejb3.server-interceptors

Server interceptor definitions.

thorntail.ejb3.statistics-enabled

If set to true, enable the collection of invocation statistics.

thorntail.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.

thorntail.ejb3.strict-max-bean-instance-pools.*KEY*.derived-size

Derived maximum number of bean instances that the pool can hold at a given point in time

thorntail.ejb3.strict-max-bean-instance-pools.*KEY*.max-pool-size

Configured maximum number of bean instances that the pool can hold at a given point in time

thorntail.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

thorntail.ejb3.strict-max-bean-instance-pools.*KEY*.timeout-unit

The instance acquisition timeout unit

thorntail.ejb3.thread-pools.*KEY*.active-count

The approximate number of threads that are actively executing tasks.

thorntail.ejb3.thread-pools.*KEY*.completed-task-count

The approximate total number of tasks that have completed execution.

thorntail.ejb3.thread-pools.*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.

thorntail.ejb3.thread-pools.*KEY*.current-thread-count

The current number of threads in the pool.

thorntail.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.

thorntail.ejb3.thread-pools.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

thorntail.ejb3.thread-pools.*KEY*.max-threads

The maximum thread pool size.

thorntail.ejb3.thread-pools.*KEY*.name

The name of the thread pool.

thorntail.ejb3.thread-pools.*KEY*.queue-size

The queue size.

thorntail.ejb3.thread-pools.*KEY*.rejected-count

The number of tasks that have been rejected.

thorntail.ejb3.thread-pools.*KEY*.task-count

The approximate total number of tasks that have ever been scheduled for execution.

thorntail.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.

thorntail.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.

thorntail.ejb3.timer-service.database-data-stores.*KEY*.database

The type of database that is in use. SQL can be customised per database type.

thorntail.ejb3.timer-service.database-data-stores.*KEY*.datasource-jndi-name

The datasource that is used to persist the timers

thorntail.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.

thorntail.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.

thorntail.ejb3.timer-service.default-data-store

The default data store used for persistent timers

thorntail.ejb3.timer-service.file-data-stores.*KEY*.path

The directory to store persistent timer information in

thorntail.ejb3.timer-service.file-data-stores.*KEY*.relative-to

The relative path that is used to resolve the timer data store location

thorntail.ejb3.timer-service.thread-pool-name

The name of the thread pool used to run timer service invocations

D.9.1. EJB MDB

Provides support for Message Driven Beans.

For this to work, you need to deploy a resource adapter for an external messaging server. The name of this resource adapter must be configured in the **ejb3** subsystem. If the resource adapter's connection factory is bound to a different JNDI name than **java:jboss/DefaultJMSConnectionFactory**, the JNDI name must be configured in the **ee** subsystem. For example:

```
thorntail:
  # deploy AMQP resource adapter
  deployment:
    org.amqphub.jca:resource-adapter.rar:
  # configure the resource adapter
  resource-adapters:
    resource-adapters:
      # the resource adapter is called `default`
      default:
        archive: resource-adapter.rar
        transaction-support: NoTransaction
        connection-definitions:
          default:
            # the connection factory is bound to JNDI name `java:global/jms/default`
            jndi-name: java:global/jms/default
            class-name: org.jboss.resource.adapter.jms.JmsManagedConnectionFactory
            config-properties:
              ConnectionFactory:
                value: factory1
              UserName:
                value: username
              Password:
                value: password
              JndiParameters:
                value:
"java.naming.factory.initial=org.apache.qpid.jms.jndi.JmsInitialContextFactory;connectionFactory.factory
=amqp://${env.MESSAGING_SERVICE_HOST:localhost}:${env.MESSAGING_SERVICE_PORT:5672
}"
  # configure the `ejb3` and `ee` subsystems
  ejb3:
    default-resource-adapter-name: default
  ee:
    annotation-property-replacement: true
    default-bindings-service:
      jms-connection-factory: java:global/jms/default
```

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>ejb-mdb</artifactId>
</dependency>
```

D.10. 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:

```
thorntail:
  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:

```
thorntail:
  elytron:
    file-audit-logs:
      local-audit:
        path: /tmp/audit.log
```

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>elytron</artifactId>
</dependency>
```

Configuration

thorntail.elytron.add-prefix-role-mappers.*KEY*.prefix

The prefix to add to each role.

thorntail.elytron.add-suffix-role-mappers.*KEY*.suffix

The suffix to add to each role.

thorntail.elytron.aggregate-evidence-decoders.*KEY*.evidence-decoders

The referenced evidence decoders to aggregate.

thorntail.elytron.aggregate-http-server-mechanism-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this factory instance.

thorntail.elytron.aggregate-http-server-mechanism-factories.*KEY*.http-server-mechanism-factories

The referenced http server factories to aggregate.

thorntail.elytron.aggregate-principal-decoders.*KEY*.principal-decoders

The referenced principal decoders to aggregate.

thorntail.elytron.aggregate-principal-transformers.*KEY*.principal-transformers

The referenced principal transformers to aggregate.

thorntail.elytron.aggregate-providers.*KEY*.providers

The referenced Provider[] resources to aggregate.

thorntail.elytron.aggregate-realms.*KEY*.authentication-realm

Reference to the security realm to use for authentication steps (obtaining or validating credentials).

thorntail.elytron.aggregate-realms.*KEY*.authorization-realm

Reference to the security realm to use for loading the identity for authorization steps (loading of the identity).

thorntail.elytron.aggregate-realms.*KEY*.authorization-realms

Reference to one or more security realms to use for loading the identity for authorization steps and aggregating the attributes (loading of the identity).

thorntail.elytron.aggregate-realms.*KEY*.principal-transformer

Reference to a principal transformer to apply between loading the identity for authentication and loading the identity for authorization

thorntail.elytron.aggregate-role-mappers.*KEY*.role-mappers

The referenced role mappers to aggregate.

thorntail.elytron.aggregate-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

thorntail.elytron.aggregate-sasl-server-factories.*KEY*.sasl-server-factories

The referenced sasl server factories to aggregate.

thorntail.elytron.aggregate-security-event-listeners.*KEY*.security-event-listeners

The referenced security event listener resources to aggregate.

thorntail.elytron.authentication-configurations.*KEY*.anonymous

Enables anonymous authentication.

thorntail.elytron.authentication-configurations.*KEY*.attribute-extends

A previously defined authentication configuration to extend.

thorntail.elytron.authentication-configurations.*KEY*.authentication-name

The authentication name to use.

thorntail.elytron.authentication-configurations.*KEY*.authorization-name

The authorization name to use.

thorntail.elytron.authentication-configurations.*KEY*.credential-reference

The reference to credential stored in CredentialStore under defined alias or clear text password.

thorntail.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.

thorntail.elytron.authentication-configurations.*KEY*.host

The host to use.

thorntail.elytron.authentication-configurations.*KEY*.kerberos-security-factory

Reference to a kerberos security factory used to obtain a GSS kerberos credential

thorntail.elytron.authentication-configurations.*KEY*.mechanism-properties

Configuration properties for the SASL authentication mechanism.

thorntail.elytron.authentication-configurations.*KEY*.port

The port to use.

thorntail.elytron.authentication-configurations.*KEY*.protocol

The protocol to use.

thorntail.elytron.authentication-configurations.*KEY*.realm

The realm to use.

thorntail.elytron.authentication-configurations.*KEY*.sasl-mechanism-selector

The SASL mechanism selector string.

thorntail.elytron.authentication-configurations.*KEY*.security-domain

Reference to a security domain to obtain a forwarded identity.

thorntail.elytron.authentication-contexts.*KEY*.attribute-extends

A previously defined authentication context to extend.

thorntail.elytron.authentication-contexts.*KEY*.match-rules

The match-rules for this authentication context.

thorntail.elytron.caching-realms.*KEY*.maximum-age

The time in milliseconds that an item can stay in the cache.

thorntail.elytron.caching-realms.*KEY*.maximum-entries

The maximum number of entries to keep in the cache.

thorntail.elytron.caching-realms.*KEY*.realm

A reference to a cacheable security realm.

thorntail.elytron.certificate-authorities.*KEY*.staging-url

URL of the certificate authority to use in pre-production.

thorntail.elytron.certificate-authorities.*KEY*.url

URL of the certificate authority.

thorntail.elytron.certificate-authority-accounts.*KEY*.alias

The alias of certificate authority account key in the keystore. If the alias does not already exist in the keystore, a certificate authority account key will be automatically generated and stored as a PrivateKeyEntry under the alias.

thorntail.elytron.certificate-authority-accounts.*KEY*.certificate-authority

The name of the certificate authority to use.

thorntail.elytron.certificate-authority-accounts.*KEY*.contact-urls

A list of URLs that the certificate authority can contact about any issues related to this account.

thorntail.elytron.certificate-authority-accounts.*KEY*.credential-reference

Credential to be used when accessing the certificate authority account key.

thorntail.elytron.certificate-authority-accounts.*KEY*.key-store

The keystore that contains the certificate authority account key.

thorntail.elytron.chained-principal-transformers.*KEY*.principal-transformers

The referenced principal transformers to chain.

thorntail.elytron.client-ssl-contexts.*KEY*.active-session-count

The count of current active sessions.

thorntail.elytron.client-ssl-contexts.*KEY*.cipher-suite-filter

The filter to apply to specify the enabled cipher suites.

thorntail.elytron.client-ssl-contexts.*KEY*.key-manager

Reference to the key manager to use within the SSLContext.

thorntail.elytron.client-ssl-contexts.*KEY*.protocols

The enabled protocols.

thorntail.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.

thorntail.elytron.client-ssl-contexts.*KEY*.providers

The name of the providers to obtain the Provider[] to use to load the SSLContext.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.application-buffer-size

The application buffer size as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.cipher-suite

The selected cipher suite as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.creation-time

The creation time as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.last-accessed-time

The last accessed time as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.local-certificates

The local certificates from the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.local-principal

The local principal as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.packet-buffer-size

The packet buffer size as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-certificates

The peer certificates from the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-host

The peer host as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-port

The peer port as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-principal

The peer principal as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.protocol

The protocol as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.ssl-sessions.*KEY*.valid

The validity of the session as reported by the SSLSession.

thorntail.elytron.client-ssl-contexts.*KEY*.trust-manager

Reference to the trust manager to use within the SSLContext.

thorntail.elytron.concatenating-principal-decoders.*KEY*.joiner

The string to use to join the results of the referenced principal decoders.

thorntail.elytron.concatenating-principal-decoders.*KEY*.principal-decoders

The referenced principal decoders to concatenate.

thorntail.elytron.configurable-http-server-mechanism-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this factory instance.

thorntail.elytron.configurable-http-server-mechanism-factories.*KEY*.filters

Filtering to be applied to enable / disable mechanisms based on the name.

thorntail.elytron.configurable-http-server-mechanism-factories.*KEY*.http-server-mechanism-factory

The http server factory to be wrapped.

thorntail.elytron.configurable-http-server-mechanism-factories.*KEY*.properties

Custom properties to be passed in to the http server factory calls.

thorntail.elytron.configurable-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

thorntail.elytron.configurable-sasl-server-factories.*KEY*.filters

List of filters to be evaluated sequentially combining the results using 'or'.

thorntail.elytron.configurable-sasl-server-factories.*KEY*.properties

Custom properties to be passed in to the sasl server factory calls.

thorntail.elytron.configurable-sasl-server-factories.*KEY*.protocol

The protocol that should be passed into factory when creating the mechanism.

thorntail.elytron.configurable-sasl-server-factories.*KEY*.sasl-server-factory

The sasl server factory to be wrapped.

thorntail.elytron.configurable-sasl-server-factories.*KEY*.server-name

The server name that should be passed into factory when creating the mechanism.

thorntail.elytron.constant-permission-mappers.*KEY*.permission-sets

The permission sets to assign.

thorntail.elytron.constant-permission-mappers.*KEY*.permissions

The permissions to assign.

thorntail.elytron.constant-principal-decoders.*KEY*.constant

The constant value the principal decoder will always return.

thorntail.elytron.constant-principal-transformers.*KEY*.constant

The constant value this PrincipalTransformer will always return.

thorntail.elytron.constant-realm-mappers.*KEY*.realm-name

The name of the constant realm to return.

thorntail.elytron.constant-role-mappers.*KEY*.roles

The constant roles to be returned by this role mapper.

thorntail.elytron.credential-stores.*KEY*.create

Specifies whether credential store should create storage when it doesn’t exist.

thorntail.elytron.credential-stores.*KEY*.credential-reference

Credential reference to be used to create protection parameter.

thorntail.elytron.credential-stores.*KEY*.implementation-properties

Map of credentials store implementation specific properties.

thorntail.elytron.credential-stores.*KEY*.location

File name of credential store storage.

thorntail.elytron.credential-stores.*KEY*.modifiable

Specifies whether credential store is modifiable.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.elytron.credential-stores.*KEY*.relative-to

A reference to a previously defined path that the file name is relative to.

thorntail.elytron.credential-stores.*KEY*.state

The state of the underlying service that represents this credential store at runtime.

thorntail.elytron.credential-stores.*KEY*.type

The credential store type, e.g. KeyStoreCredentialStore.

thorntail.elytron.custom-credential-security-factories.*KEY*.class-name

The class name of the implementation of the custom security factory.

thorntail.elytron.custom-credential-security-factories.*KEY*.configuration

The optional key/value configuration for the custom security factory.

thorntail.elytron.custom-credential-security-factories.*KEY*.module

The module to use to load the custom security factory.

thorntail.elytron.custom-evidence-decoders.*KEY*.class-name

Fully qualified class name of the evidence decoder.

thorntail.elytron.custom-evidence-decoders.*KEY*.configuration

The optional key/value configuration for the evidence decoder.

thorntail.elytron.custom-evidence-decoders.*KEY*.module

Name of the module to use to load the evidence decoder.

thorntail.elytron.custom-modifiable-realms.*KEY*.class-name

The class name of the implementation of the custom realm.

thorntail.elytron.custom-modifiable-realms.*KEY*.configuration

The optional key/value configuration for the custom realm.

thorntail.elytron.custom-modifiable-realms.*KEY*.module

The module to use to load the custom realm.

thorntail.elytron.custom-permission-mappers.*KEY*.class-name

Fully qualified class name of the permission mapper

thorntail.elytron.custom-permission-mappers.*KEY*.configuration

The optional kay/value configuration for the permission mapper

thorntail.elytron.custom-permission-mappers.*KEY*.module

Name of the module to use to load the permission mapper

thorntail.elytron.custom-principal-decoders.*KEY*.class-name

Fully qualified class name of the principal decoder

thorntail.elytron.custom-principal-decoders.*KEY*.configuration

The optional kay/value configuration for the principal decoder

thorntail.elytron.custom-principal-decoders.*KEY*.module

Name of the module to use to load the principal decoder

thorntail.elytron.custom-principal-transformers.*KEY*.class-name

The class name of the implementation of the custom principal transformer.

thorntail.elytron.custom-principal-transformers.*KEY*.configuration

The optional key/value configuration for the custom principal transformer.

thorntail.elytron.custom-principal-transformers.*KEY*.module

The module to use to load the custom principal transformer.

thorntail.elytron.custom-realm-mappers.*KEY*.class-name

Fully qualified class name of the RealmMapper

thorntail.elytron.custom-realm-mappers.*KEY*.configuration

The optional kay/value configuration for the RealmMapper

thorntail.elytron.custom-realm-mappers.*KEY*.module

Name of the module to use to load the RealmMapper

thorntail.elytron.custom-realms.*KEY*.class-name

The class name of the implementation of the custom realm.

thorntail.elytron.custom-realms.*KEY*.configuration

The optional key/value configuration for the custom realm.

thorntail.elytron.custom-realms.*KEY*.module

The module to use to load the custom realm.

thorntail.elytron.custom-role-decoders.*KEY*.class-name

Fully qualified class name of the RoleDecoder

thorntail.elytron.custom-role-decoders.*KEY*.configuration

The optional kay/value configuration for the RoleDecoder

thorntail.elytron.custom-role-decoders.*KEY*.module

Name of the module to use to load the RoleDecoder

thorntail.elytron.custom-role-mappers.*KEY*.class-name

Fully qualified class name of the RoleMapper

thorntail.elytron.custom-role-mappers.*KEY*.configuration

The optional key/value configuration for the RoleMapper

thorntail.elytron.custom-role-mappers.*KEY*.module

Name of the module to use to load the RoleMapper

thorntail.elytron.custom-security-event-listeners.*KEY*.class-name

The class name of the implementation of the custom security event listener.

thorntail.elytron.custom-security-event-listeners.*KEY*.configuration

The optional key/value configuration for the custom security event listener.

thorntail.elytron.custom-security-event-listeners.*KEY*.module

The module to use to load the custom security event listener.

thorntail.elytron.default-authentication-context

The default authentication context to be associated with all deployments.

thorntail.elytron.default-ssl-context

Reference to the SSLContext which should be globally registered as the default.

thorntail.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).

thorntail.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.

thorntail.elytron.dir-contexts.*KEY*.connection-timeout

The timeout for connecting to the LDAP server in milliseconds.

thorntail.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).

thorntail.elytron.dir-contexts.*KEY*.enable-connection-pooling

Indicates if connection pooling is enabled.

thorntail.elytron.dir-contexts.*KEY*.module

Name of module that will be used as class loading base.

thorntail.elytron.dir-contexts.*KEY*.principal

The principal to authenticate and connect to the LDAP server. Can be omitted if authentication-level is 'none' (anonymous).

thorntail.elytron.dir-contexts.*KEY*.properties

The additional connection properties for the DirContext.

thorntail.elytron.dir-contexts.*KEY*.read-timeout

The read timeout for an LDAP operation in milliseconds.

thorntail.elytron.dir-contexts.*KEY*.referral-mode

If referrals should be followed.

thorntail.elytron.dir-contexts.*KEY*.ssl-context

The name of ssl-context used to secure connection to the LDAP server.

thorntail.elytron.dir-contexts.*KEY*.url

The connection url.

thorntail.elytron.disallowed-providers

A list of providers that are not allowed, and will be removed from the providers list.

thorntail.elytron.file-audit-logs.*KEY*.attribute-synchronized

Whether every event should be immediately synchronised to disk.

thorntail.elytron.file-audit-logs.*KEY*.autoflush

Whether every event should be immediately flushed to disk (If undefined will default to the value of synchronized).

thorntail.elytron.file-audit-logs.*KEY*.format

The format to use to record the audit event.

thorntail.elytron.file-audit-logs.*KEY*.path

Path of the file to be written.

thorntail.elytron.file-audit-logs.*KEY*.relative-to

The relative path to the audit log.

thorntail.elytron.filesystem-realms.*KEY*.encoded

Whether the identity names should be stored encoded (Base32) in file names.

thorntail.elytron.filesystem-realms.*KEY*.levels

The number of levels of directory hashing to apply.

thorntail.elytron.filesystem-realms.*KEY*.path

The path to the file containing the realm.

thorntail.elytron.filesystem-realms.*KEY*.relative-to

The pre-defined path the path is relative to.

thorntail.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

thorntail.elytron.filtering-key-stores.*KEY*.key-store

Name of filtered KeyStore.

thorntail.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.

thorntail.elytron.final-providers

Reference to the Providers that should be registered after all existing Providers.

thorntail.elytron.http-authentication-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this configuration after all filtering has been applied.

thorntail.elytron.http-authentication-factories.*KEY*.http-server-mechanism-factory

The HttpServerAuthenticationMechanismFactory to associate with this resource

thorntail.elytron.http-authentication-factories.*KEY*.mechanism-configurations

Mechanism specific configuration

thorntail.elytron.http-authentication-factories.*KEY*.security-domain

The SecurityDomain to associate with this resource

thorntail.elytron.identity-realms.*KEY*.attribute-name

The name of the attribute associated with this identity.

thorntail.elytron.identity-realms.*KEY*.attribute-values

The values associated with the identity attributes.

thorntail.elytron.identity-realms.*KEY*.identity

The name of the identity available from the security realm.

thorntail.elytron.initial-providers

Reference to the Providers that should be registered ahead of all existing Providers.

thorntail.elytron.jaspi-configurations.*KEY*.application-context

The application-context this configuration is to be associated with or '*' for all application contexts.

thorntail.elytron.jaspi-configurations.*KEY*.description

Description of this configuration.

thorntail.elytron.jaspi-configurations.*KEY*.layer

The layer this configuration is to be associated with or '*' for all layers.

thorntail.elytron.jaspi-configurations.*KEY*.server-auth-modules

List of ServerAuthModule definitions for this configuration.

thorntail.elytron.jdbc-realms.*KEY*.principal-query

The authentication query used to authenticate users based on specific key types.

thorntail.elytron.kerberos-security-factories.*KEY*.debug

Should the JAAS step of obtaining the credential have debug logging enabled.

thorntail.elytron.kerberos-security-factories.*KEY*.fail-cache

Amount of seconds before new try to obtain server credential should be done if it has failed last time.

thorntail.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.

thorntail.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.

thorntail.elytron.kerberos-security-factories.*KEY*.minimum-remaining-lifetime

How much lifetime (in seconds) should a cached credential have remaining before it is recreated.

thorntail.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.

thorntail.elytron.kerberos-security-factories.*KEY*.options

The Krb5LoginModule additional options.

thorntail.elytron.kerberos-security-factories.*KEY*.path

The path of the KeyTab to load to obtain the credential.

thorntail.elytron.kerberos-security-factories.*KEY*.principal

The principal represented by the KeyTab

thorntail.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.

thorntail.elytron.kerberos-security-factories.*KEY*.request-lifetime

How much lifetime (in seconds) should be requested for newly created credentials.

thorntail.elytron.kerberos-security-factories.*KEY*.required

Is the keytab file with adequate principal required to exist at the time the service starts?

thorntail.elytron.kerberos-security-factories.*KEY*.server

If this for use server side or client side?

thorntail.elytron.kerberos-security-factories.*KEY*.wrap-gss-credential

Should generated GSS credentials be wrapped to prevent improper disposal or not?

thorntail.elytron.key-managers.*KEY*.algorithm

The name of the algorithm to use to create the underlying KeyManagerFactory.

thorntail.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

thorntail.elytron.key-managers.*KEY*.credential-reference

The credential reference to decrypt KeyStore item. (Not a password of the KeyStore.)

thorntail.elytron.key-managers.*KEY*.key-store

Reference to the KeyStore to use to initialise the underlying KeyManagerFactory.

thorntail.elytron.key-managers.*KEY*.provider-name

The name of the provider to use to create the underlying KeyManagerFactory.

thorntail.elytron.key-managers.*KEY*.providers

Reference to obtain the Provider[] to use when creating the underlying KeyManagerFactory.

thorntail.elytron.key-store-realms.*KEY*.key-store

Reference to the KeyStore that should be used to back this security realm.

thorntail.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

thorntail.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.

thorntail.elytron.key-stores.*KEY*.credential-reference

The reference to credential stored in CredentialStore under defined alias or clear text password.

thorntail.elytron.key-stores.*KEY*.loaded-provider

Information about the provider that was used for this KeyStore.

thorntail.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.

thorntail.elytron.key-stores.*KEY*.path

The path to the KeyStore file.

thorntail.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.

thorntail.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.

thorntail.elytron.key-stores.*KEY*.relative-to

The base path this store is relative to.

thorntail.elytron.key-stores.*KEY*.required

Is the file required to exist at the time the KeyStore service starts?

thorntail.elytron.key-stores.*KEY*.size

The number of entries in the KeyStore.

thorntail.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.

thorntail.elytron.key-stores.*KEY*.type

The type of the KeyStore, used when creating the new KeyStore instance.

thorntail.elytron.ldap-key-stores.*KEY*.alias-attribute

The name of LDAP attribute, where will be item alias stored.

thorntail.elytron.ldap-key-stores.*KEY*.certificate-attribute

The name of LDAP attribute, where will be certificate stored.

thorntail.elytron.ldap-key-stores.*KEY*.certificate-chain-attribute

The name of LDAP attribute, where will be certificate chain stored.

thorntail.elytron.ldap-key-stores.*KEY*.certificate-chain-encoding

The encoding of the certificate chain.

thorntail.elytron.ldap-key-stores.*KEY*.certificate-type

The type of the Certificate.

thorntail.elytron.ldap-key-stores.*KEY*.dir-context

The name of DirContext, which will be used to communication with LDAP server.

thorntail.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'.

thorntail.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'.

thorntail.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'.

thorntail.elytron.ldap-key-stores.*KEY*.key-attribute

The name of LDAP attribute, where will be key stored.

thorntail.elytron.ldap-key-stores.*KEY*.key-type

The type of KeyStore, in which will be key serialized to LDAP attribute.

thorntail.elytron.ldap-key-stores.*KEY*.new-item-template

Configuration for item creation. Define how will look LDAP entry of newly created keystore item.

thorntail.elytron.ldap-key-stores.*KEY*.search-path

The path in LDAP, where will be KeyStore items searched.

thorntail.elytron.ldap-key-stores.*KEY*.search-recursive

If the LDAP search should be recursive.

thorntail.elytron.ldap-key-stores.*KEY*.search-time-limit

The time limit for obtaining keystore items from LDAP.

thorntail.elytron.ldap-key-stores.*KEY*.size

The size of LDAP KeyStore in amount of items/aliases.

thorntail.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.

thorntail.elytron.ldap-realms.*KEY*.allow-blank-password

Does this realm support blank password direct verification? Blank password attempt will be rejected otherwise.

thorntail.elytron.ldap-realms.*KEY*.dir-context

The configuration to connect to a LDAP server.

thorntail.elytron.ldap-realms.*KEY*.direct-verification

Does this realm support verification of credentials by directly connecting to LDAP as the account being authenticated?

thorntail.elytron.ldap-realms.*KEY*.identity-mapping

The configuration options that define how principals are mapped to their corresponding entries in the underlying LDAP server.

thorntail.elytron.logical-permission-mappers.*KEY*.left

Reference to the permission mapper to use to the left of the operation.

thorntail.elytron.logical-permission-mappers.*KEY*.logical-operation

The logical operation to use to combine the permission mappers.

thorntail.elytron.logical-permission-mappers.*KEY*.right

Reference to the permission mapper to use to the right of the operation.

thorntail.elytron.logical-role-mappers.*KEY*.left

Reference to a role mapper to be used on the left side of the operation.

thorntail.elytron.logical-role-mappers.*KEY*.logical-operation

The logical operation to be performed on the role mapper mappings.

thorntail.elytron.logical-role-mappers.*KEY*.right

Reference to a role mapper to be used on the right side of the operation.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.elytron.mapped-role-mappers.*KEY*.keep-mapped

When set to 'true' the mapped roles will retain all roles, that have defined mappings.

thorntail.elytron.mapped-role-mappers.*KEY*.keep-non-mapped

When set to 'true' the mapped roles will retain all roles, that have no defined mappings.

thorntail.elytron.mapped-role-mappers.*KEY*.role-map

A string to string list map for mapping roles.

thorntail.elytron.mechanism-provider-filtering-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

thorntail.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.

thorntail.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.

thorntail.elytron.mechanism-provider-filtering-sasl-server-factories.*KEY*.sasl-server-factory

Reference to a sasl server factory to be wrapped by this definition.

thorntail.elytron.periodic-rotating-file-audit-logs.*KEY*.attribute-synchronized

Whether every event should be immediately synchronised to disk.

thorntail.elytron.periodic-rotating-file-audit-logs.*KEY*.autoflush

Whether every event should be immediately flushed to disk (If undefined will default to the value of synchronized).

thorntail.elytron.periodic-rotating-file-audit-logs.*KEY*.format

The format to use to record the audit event.

thorntail.elytron.periodic-rotating-file-audit-logs.*KEY*.path

Path of the file to be written.

thorntail.elytron.periodic-rotating-file-audit-logs.*KEY*.relative-to

The relative path to the audit log.

thorntail.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.

thorntail.elytron.permission-sets.*KEY*.permissions

The permissions in the permission set.

thorntail.elytron.policies.*KEY*.custom-policy

A custom policy provider definition.

thorntail.elytron.policies.*KEY*.default-policy

Not used.

thorntail.elytron.policies.*KEY*.jacc-policy

A policy provider definition that sets up JACC and related services.

thorntail.elytron.properties-realms.*KEY*.attribute-synchronized

The time the properties files that back this realm were last loaded.

thorntail.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.

thorntail.elytron.properties-realms.*KEY*.groups-properties

The properties file containing the users and their groups.

thorntail.elytron.properties-realms.*KEY*.users-properties

The properties file containing the users and their passwords.

thorntail.elytron.provider-http-server-mechanism-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this factory instance.

thorntail.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.

thorntail.elytron.provider-loaders.*KEY*.argument

An argument to be passed into the constructor as the Provider is instantiated.

thorntail.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.

thorntail.elytron.provider-loaders.*KEY*.configuration

The key/value configuration to be passed to the Provider to initialise it.

thorntail.elytron.provider-loaders.*KEY*.loaded-providers

The list of providers loaded by this provider loader.

thorntail.elytron.provider-loaders.*KEY*.module

The name of the module to load the provider from.

thorntail.elytron.provider-loaders.*KEY*.path

The path of the file to use to initialise the providers.

thorntail.elytron.provider-loaders.*KEY*.relative-to

The base path of the configuration file.

thorntail.elytron.provider-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

thorntail.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.

thorntail.elytron.regex-principal-transformers.*KEY*.pattern

The regular expression to use to locate the portion of the name to be replaced.

thorntail.elytron.regex-principal-transformers.*KEY*.replace-all

Should all occurrences of the pattern matched be replaced or only the first occurrence.

thorntail.elytron.regex-principal-transformers.*KEY*.replacement

The value to be used as the replacement.

thorntail.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.

thorntail.elytron.regex-validating-principal-transformers.*KEY*.pattern

The regular expression to use for the principal transformer.

thorntail.elytron.register-jaspi-factory

Should the Elytron JASPI factory be globally registered?

thorntail.elytron.sasl-authentication-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this configuration after all filtering has been applied.

thorntail.elytron.sasl-authentication-factories.*KEY*.mechanism-configurations

Mechanism specific configuration

thorntail.elytron.sasl-authentication-factories.*KEY*.sasl-server-factory

The SaslServerFactory to associate with this resource

thorntail.elytron.sasl-authentication-factories.*KEY*.security-domain

The SecurityDomain to associate with this resource

thorntail.elytron.security-domains.*KEY*.default-realm

The default realm contained by this security domain.

thorntail.elytron.security-domains.*KEY*.evidence-decoder

A reference to an EvidenceDecoder to be used by this domain.

thorntail.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.

thorntail.elytron.security-domains.*KEY*.outflow-security-domains

The list of security domains that the security identity from this domain should automatically outflow to.

thorntail.elytron.security-domains.*KEY*.permission-mapper

A reference to a PermissionMapper to be used by this domain.

thorntail.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.

thorntail.elytron.security-domains.*KEY*.pre-realm-principal-transformer

A reference to a principal transformer to be applied before the realm is selected.

thorntail.elytron.security-domains.*KEY*.principal-decoder

A reference to a PrincipalDecoder to be used by this domain.

thorntail.elytron.security-domains.*KEY*.realm-mapper

Reference to the RealmMapper to be used by this domain.

thorntail.elytron.security-domains.*KEY*.realms

The list of realms contained by this security domain.

thorntail.elytron.security-domains.*KEY*.role-mapper

Reference to the RoleMapper to be used by this domain.

thorntail.elytron.security-domains.*KEY*.security-event-listener

Reference to a listener for security events.

thorntail.elytron.security-domains.*KEY*.trusted-security-domains

The list of security domains that are trusted by this security domain.

thorntail.elytron.security-properties

Security properties to be set.

thorntail.elytron.server-ssl-contexts.*KEY*.active-session-count

The count of current active sessions.

thorntail.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.

thorntail.elytron.server-ssl-contexts.*KEY*.cipher-suite-filter

The filter to apply to specify the enabled cipher suites.

thorntail.elytron.server-ssl-contexts.*KEY*.final-principal-transformer

A final principal transformer to apply for this mechanism realm.

thorntail.elytron.server-ssl-contexts.*KEY*.key-manager

Reference to the key manager to use within the SSLContext.

thorntail.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.

thorntail.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.

thorntail.elytron.server-ssl-contexts.*KEY*.post-realm-principal-transformer

A principal transformer to apply after the realm is selected.

thorntail.elytron.server-ssl-contexts.*KEY*.pre-realm-principal-transformer

A principal transformer to apply before the realm is selected.

thorntail.elytron.server-ssl-contexts.*KEY*.protocols

The enabled protocols.

thorntail.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.

thorntail.elytron.server-ssl-contexts.*KEY*.providers

The name of the providers to obtain the Provider[] to use to load the SSLContext.

thorntail.elytron.server-ssl-contexts.*KEY*.realm-mapper

The realm mapper to be used for SSL authentication.

thorntail.elytron.server-ssl-contexts.*KEY*.security-domain

The security domain to use for authentication during SSL session establishment.

thorntail.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.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.application-buffer-size

The application buffer size as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.cipher-suite

The selected cipher suite as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.creation-time

The creation time as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.last-accessed-time

The last accessed time as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.local-certificates

The local certificates from the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.local-principal

The local principal as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.packet-buffer-size

The packet buffer size as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-certificates

The peer certificates from the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-host

The peer host as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-port

The peer port as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.peer-principal

The peer principal as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.protocol

The protocol as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.ssl-sessions.*KEY*.valid

The validity of the session as reported by the SSLSession.

thorntail.elytron.server-ssl-contexts.*KEY*.trust-manager

Reference to the trust manager to use within the SSLContext.

thorntail.elytron.server-ssl-contexts.*KEY*.use-cipher-suites-order

To honor local cipher suites preference.

thorntail.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.

thorntail.elytron.server-ssl-contexts.*KEY*.wrap

Should the SSLEngine, SSLSocket, and SSLServerSocket instances returned be wrapped to protect against further modification.

thorntail.elytron.server-ssl-sni-contexts.*KEY*.default-ssl-context

The context to use if no SNI information is present, or if it does not match any mappings

thorntail.elytron.server-ssl-sni-contexts.*KEY*.host-context-map

A mapping between a server name and an SSContext

thorntail.elytron.service-loader-http-server-mechanism-factories.*KEY*.available-mechanisms

The HTTP mechanisms available from this factory instance.

thorntail.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.

thorntail.elytron.service-loader-sasl-server-factories.*KEY*.available-mechanisms

The SASL mechanisms available from this factory after all filtering has been applied.

thorntail.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.

thorntail.elytron.simple-permission-mappers.*KEY*.mapping-mode

The mapping mode that should be used in the event of multiple matches.

thorntail.elytron.simple-permission-mappers.*KEY*.permission-mappings

The defined permission mappings.

thorntail.elytron.simple-regex-realm-mappers.*KEY*.delegate-realm-mapper

The RealmMapper to delegate to if there is no match using the pattern.

thorntail.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.

thorntail.elytron.simple-role-decoders.*KEY*.attribute

The name of the attribute from the identity to map directly to roles.

thorntail.elytron.size-rotating-file-audit-logs.*KEY*.attribute-synchronized

Whether every event should be immediately flushed to disk (If undefined will default to the value of synchronized).

thorntail.elytron.size-rotating-file-audit-logs.*KEY*.autoflush

Whether every event should be immediately flushed to disk.

thorntail.elytron.size-rotating-file-audit-logs.*KEY*.format

The format to use to record the audit event.

thorntail.elytron.size-rotating-file-audit-logs.*KEY*.max-backup-index

The maximum number of files to backup when rotating.

thorntail.elytron.size-rotating-file-audit-logs.*KEY*.path

Path of the file to be written.

thorntail.elytron.size-rotating-file-audit-logs.*KEY*.relative-to

The relative path to the audit log.

thorntail.elytron.size-rotating-file-audit-logs.*KEY*.rotate-on-boot

Whether the file should be rotated before the a new file is set.

thorntail.elytron.size-rotating-file-audit-logs.*KEY*.rotate-size

The log file size the file should rotate at.

thorntail.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.

thorntail.elytron.syslog-audit-logs.*KEY*.format

The format to use to record the audit event.

thorntail.elytron.syslog-audit-logs.*KEY*.host-name

The host name to embed withing all events sent to the remote syslog server.

thorntail.elytron.syslog-audit-logs.*KEY*.port

The listening port on the syslog server.

thorntail.elytron.syslog-audit-logs.*KEY*.reconnect-attempts

The maximum amount of failed reconnect attempts that should be made for sending messages to a syslog server before the endpoint is closed.

thorntail.elytron.syslog-audit-logs.*KEY*.server-address

The server address of the syslog server the events should be sent to.

thorntail.elytron.syslog-audit-logs.*KEY*.ssl-context

The SSLContext to use to connect to the syslog server when SSL_TCP transport is used.

thorntail.elytron.syslog-audit-logs.*KEY*.syslog-format

The RFC format to be used for describing the audit event.

thorntail.elytron.syslog-audit-logs.*KEY*.transport

The transport to use to connect to the syslog server.

thorntail.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.

thorntail.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).

thorntail.elytron.token-realms.*KEY*.principal-claim

The name of the claim that should be used to obtain the principal's name.

thorntail.elytron.trust-managers.*KEY*.algorithm

The name of the algorithm to use to create the underlying TrustManagerFactory.

thorntail.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

thorntail.elytron.trust-managers.*KEY*.certificate-revocation-list

Enables certificate revocation list checks to a trust manager.

thorntail.elytron.trust-managers.*KEY*.key-store

Reference to the KeyStore to use to initialise the underlying TrustManagerFactory.

thorntail.elytron.trust-managers.*KEY*.maximum-cert-path

The maximum number of non-self-issued intermediate certificates that may exist in a certification path for OCSP and CRL checks. If neither OCSP and CRL is configured, this attribute has no effect.

thorntail.elytron.trust-managers.*KEY*.ocsp

Enables online certificate status protocol checks to a trust manager.

thorntail.elytron.trust-managers.*KEY*.only-leaf-cert

Whether only leaf certificate should be checked for revocation status.

thorntail.elytron.trust-managers.*KEY*.provider-name

The name of the provider to use to create the underlying TrustManagerFactory.

thorntail.elytron.trust-managers.*KEY*.providers

Reference to obtain the Provider[] to use when creating the underlying TrustManagerFactory.

thorntail.elytron.trust-managers.*KEY*.soft-fail

Whether a certificate with unknown OCSP response should be accepted.

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.attribute-name

The name of the X.500 attribute to map (can be defined using OID instead)

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.convert

When set to 'true', if the Principal is not already an X500Principal conversion will be attempted

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.joiner

The joining string

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.maximum-segments

The maximum number of occurrences of the attribute to map

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.oid

The OID of the X.500 attribute to map (can be defined using attribute name instead)

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.required-attributes

The attributes names of the attributes that must be present in the principal

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.required-oids

The OIDs of the attributes that must be present in the principal

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.reverse

When set to 'true', the attribute values will be processed and returned in reverse order

thorntail.elytron.x500-attribute-principal-decoders.*KEY*.start-segment

The 0-based starting occurrence of the attribute to map

thorntail.elytron.x509-subject-alt-name-evidence-decoders.*KEY*.alt-name-type

The subject alternative name type to decode from the given evidence. Allowed values: 'rfc822Name', 'dNSName', 'directoryName', 'uniformResourceIdentifier', 'iPAddress', 'registeredID'

thorntail.elytron.x509-subject-alt-name-evidence-decoders.*KEY*.segment

The 0-based occurrence of the subject alternative name to map. This attribute is optional and only used when there is more than one subject alternative name of the given alt-name-type. The default value is 0.

D.11. HIBERNATE

This fraction provides the same functionality as the **jpa** fraction.

Additionally, the **hibernate** fraction provides dependencies on the correct versions of Hibernate artifacts. If you use Hibernate APIs in your applications, you can use this fraction to manage the Hibernate versions instead of managing them manually.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>hibernate</artifactId>
</dependency>
```

D.11.1. Hibernate Validator

Provides support and integration for applications using Hibernate Validator.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>hibernate-validator</artifactId>
</dependency>
```

D.12. HYSTRIX

WARNING

This fraction is deprecated.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>hystrix</artifactId>
</dependency>
```

Configuration

thorntail.hystrix.collapser.default.maxRequestsInBatch

The maximum number of requests allowed in a batch before this triggers a batch execution

thorntail.hystrix.collapser.default.requestCache.enabled

Indicates whether request caching is enabled for HystrixCollapser.execute() and HystrixCollapser.queue() invocations

thorntail.hystrix.collapser.default.timerDelayInMilliseconds

The number of milliseconds after the creation of the batch that its execution is triggered

thorntail.hystrix.command.default.circuitBreaker.enabled

Determines whether a circuit breaker will be used to track health and to short-circuit requests if it trips

thorntail.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

thorntail.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

thorntail.hystrix.command.default.circuitBreaker.forceOpen

If true, forces the circuit breaker into an open (tripped) state in which it will reject all requests

thorntail.hystrix.command.default.circuitBreaker.requestVolumeThreshold

The minimum number of requests in a rolling window that will trip the circuit

thorntail.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

thorntail.hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests

The maximum number of requests allowed to a HystrixCommand.run() method when you are using ExecutionIsolationStrategy.SEMAPHORE

thorntail.hystrix.command.default.execution.isolation.strategy

Isolation strategy (THREAD or SEMAPHORE)

thorntail.hystrix.command.default.execution.isolation.thread.interruptOnCancel

Indicates whether the HystrixCommand.run() execution should be interrupted when a cancellation occurs

thorntail.hystrix.command.default.execution.isolation.thread.interruptOnTimeout

Indicates whether the HystrixCommand.run() execution should be interrupted when a timeout occurs

thorntail.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

thorntail.hystrix.command.default.execution.timeout.enabled

Indicates whether the HystrixCommand.run() execution should have a timeout

thorntail.hystrix.command.default.fallback.enabled

Determines whether a call to HystrixCommand.getFallback() will be attempted when failure or rejection occurs

thorntail.hystrix.command.default.fallback.isolation.semaphore.maxConcurrentRequests

The maximum number of requests allowed to a HystrixCommand.getFallback() method when you are using ExecutionIsolationStrategy.SEMAPHORE

thorntail.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

thorntail.hystrix.command.default.metrics.rollingPercentile.bucketSize

The maximum number of execution times that are kept per bucket

thorntail.hystrix.command.default.metrics.rollingPercentile.enabled

Indicates whether execution latencies should be tracked and calculated as percentiles

thorntail.hystrix.command.default.metrics.rollingPercentile.numBuckets

The number of buckets the rollingPercentile window will be divided into

thorntail.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

thorntail.hystrix.command.default.metrics.rollingStats.numBuckets

The number of buckets the rolling statistical window is divided into

thorntail.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

thorntail.hystrix.command.default.requestCache.enabled

Indicates whether HystrixCommand.getCacheKey() should be used with HystrixRequestCache to provide de-duplication functionality via request-scoped caching

thorntail.hystrix.command.default.requestLog.enabled

Indicates whether HystrixCommand execution and events should be logged to HystrixRequestLog

thorntail.hystrix.stream.path

Context path for the stream

thorntail.hystrix.threadpool.default.allowMaximumSizeToDivergeFromCoreSize

Allows the configuration for maximumSize to take effect

thorntail.hystrix.threadpool.default.coreSize

The core thread-pool size

thorntail.hystrix.threadpool.default.keepAliveTimeMinutes

The keep-alive time, in minutes

thorntail.hystrix.threadpool.default.maxQueueSize

The maximum queue size of the BlockingQueue implementation

thorntail.hystrix.threadpool.default.maximumSize

The maximum thread-pool size

thorntail.hystrix.threadpool.default.metrics.rollingPercentile.numBuckets

The number of buckets the rolling statistical window is divided into

thorntail.hystrix.threadpool.default.metrics.rollingStats.timeInMilliseconds

The duration of the statistical rolling window, in milliseconds

thorntail.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.13. INFINISPAN

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>infinispan</artifactId>
</dependency>
```

Configuration

thorntail.infinispan.cache-containers.*KEY*.aliases

The list of aliases for this cache container

thorntail.infinispan.cache-containers.*KEY*.async-operations-thread-pool.keepalive-time

Used to specify the amount of milliseconds that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

thorntail.infinispan.cache-containers.*KEY*.async-operations-thread-pool.max-threads

The maximum thread pool size.

thorntail.infinispan.cache-containers.*KEY*.async-operations-thread-pool.min-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.

thorntail.infinispan.cache-containers.*KEY*.async-operations-thread-pool.queue-length

The queue length.

thorntail.infinispan.cache-containers.*KEY*.cache-manager-status

The status of the cache manager component. May return null if the cache manager is not started.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.activations

The number of cache node activations (bringing a node into memory from a cache store).

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.average-read-time

Average time (in ms) for cache reads. Includes hits and misses.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.average-remove-time

Average time (in ms) for cache removes.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.average-replication-time

The average time taken to replicate data around the cluster.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.average-write-time

Average time (in ms) for cache writes.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.hit-ratio

The hit/miss ratio for the cache (hits/hits+misses).

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.hits

The number of cache attribute hits.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.invalidations

The number of cache invalidations.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.locking-component.current-concurrency-level

The estimated number of concurrently updating threads which this cache can support.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.locking-component.number-of-locks-available

The number of locks available to this cache.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.locking-component.number-of-locks-held

The number of locks currently in use by this cache.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.misses

The number of cache attribute misses.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.number-of-entries

The number of entries in the cache including passivated entries.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.number-of-entries-in-memory

The number of entries in the cache excluding passivated entries.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.partition-handling-component.availability

Indicates the current availability of the cache.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.passivations

The number of cache node passivations (passivating a node from memory to a cache store).

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.persistence-component.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.persistence-component.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.read-write-ratio

The read/write ratio of the cache ((hits+misses)/stores).

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.remove-hits

The number of cache attribute remove hits.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.remove-misses

The number of cache attribute remove misses.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.replication-count

The number of times data was replicated around the cluster.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.replication-failures

The number of data replication failures.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.success-ratio

The data replication success ratio (successes/successes+failures).

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.time-since-reset

Time (in secs) since cache statistics were reset.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.time-since-start

Time (in secs) since cache was started.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.transaction-component.commits

The number of transaction commits.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.transaction-component.prepares

The number of transaction prepares.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.transaction-component.rollbacks

The number of transaction rollbacks.

thorntail.infinispan.cache-containers.*KEY*.caches.*KEY*.writes

The number of cache attribute put operations.

thorntail.infinispan.cache-containers.*KEY*.cluster-name

The name of the cluster this node belongs to. May return null if the cache manager is not started.

thorntail.infinispan.cache-containers.*KEY*.coordinator-address

The logical address of the cluster’s coordinator. May return null if the cache manager is not started.

thorntail.infinispan.cache-containers.*KEY*.default-cache

The default infinispan cache

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.activations

The number of cache node activations (bringing a node into memory from a cache store).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.async-marshalling

If enabled, this will cause marshalling of entries to be performed asynchronously.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.average-read-time

Average time (in ms) for cache reads. Includes hits and misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.average-replication-time

The average time taken to replicate data around the cluster.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.average-write-time

Average time (in ms) for cache writes.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.backup-for-component.remote-cache

The name of the remote cache for which this cache acts as a backup.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.backup-for-component.remote-site

The site of the remote cache for which this cache acts as a backup.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.backups-component.backups.*KEY*.after-failures

Indicates the number of failures after which this backup site should go offline.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.backups-component.backups.*KEY*.enabled

Indicates whether or not this backup site is enabled.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.backups-component.backups.*KEY*.failure-policy

The policy to follow when connectivity to the backup site fails.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.backups-component.backups.*KEY*.min-wait

Indicates the minimum time (in milliseconds) to wait after the max number of failures is reached, after which this backup site should go offline.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.backups-component.backups.*KEY*.strategy

The backup strategy for this cache

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.backups-component.backups.*KEY*.timeout

The timeout for replicating to the backup site.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.batching

If enabled, the invocation batching API will be made available for this cache.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.binary-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.cache-status

The status of the cache component.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.capacity-factor

Controls the proportion of entries that will reside on the local node, compared to the other nodes in the cluster.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.consistent-hash-strategy

Defines the consistent hash strategy for the cache.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.attribute-class

The custom store implementation class to use for this cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.custom-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.elapsed-time

Time (in secs) since cache started.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.expiration-component.interval

Interval (in milliseconds) between subsequent runs to purge expired entries from memory and any cache stores. If you wish to disable the periodic eviction process altogether, set wakeupInterval to -1.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.expiration-component.lifespan

Maximum lifespan of a cache entry, after which the entry is expired cluster-wide, in milliseconds. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.expiration-component.max-idle

Maximum idle time a cache entry will be maintained in the cache, in milliseconds. If the idle time is exceeded, the entry will be expired cluster-wide. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.path

The system path under which this cache store will persist its entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.relative-to

The system path to which the specified path is relative.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.file-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hit-ratio

The hit/miss ratio for the cache (hits/hits+misses).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hits

The number of cache attribute hits.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.cache-configuration

Name of the cache configuration template defined in Infinispan Server to create caches from.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.remote-cache-container

Reference to a container-managed remote-cache-container.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.hotrod-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.indexing

If enabled, entries will be indexed when they are added to the cache. Indexes will be updated as entries change or are removed.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.indexing-properties

Properties to control indexing behaviour

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.invalidations

The number of cache invalidations.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.jndi-name

The jndi-name to which to bind this cache instance.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.l1-lifespan

Maximum lifespan of an entry placed in the L1 cache. This element configures the L1 cache behavior in 'distributed' caches instances. In any other cache modes, this element is ignored.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.locking-component.acquire-timeout

Maximum time to attempt a particular lock acquisition.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.locking-component.concurrency-level

Concurrency level for lock containers. Adjust this value according to the number of concurrent threads interacting with Infinispan.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.locking-component.current-concurrency-level

The estimated number of concurrently updating threads which this cache can support.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.locking-component.isolation

Sets the cache locking isolation level.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.locking-component.number-of-locks-available

The number of locks available to this cache.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.locking-component.number-of-locks-held

The number of locks currently in use by this cache.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.locking-component.striping

If true, a pool of shared locks is maintained for all entries that need to be locked. Otherwise, a lock is created per entry in the cache. Lock striping helps control memory footprint but may reduce concurrency in the system.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.misses

The number of cache attribute misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mixed-jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.mode

Sets the clustered cache mode, ASYNC for asynchronous operation, or SYNC for synchronous operation.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.module

The module whose class loader should be used when building this cache's configuration.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.number-of-entries

The number of entries in the cache including passivated entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.object-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.object-memory.max-entries

Maximum number of entries in a cache instance. If selected value is not a power of two the actual value will default to the least power of two larger than selected value. -1 means no limit.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.object-memory.size

Triggers eviction of the least recently used entries when the number of cache entries exceeds this threshold.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.object-memory.strategy

Sets the cache eviction strategy. Available options are 'UNORDERED', 'FIFO', 'LRU', 'LIRS' and 'NONE' (to disable eviction).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.off-heap-memory.capacity

Defines the capacity of the off-heap storage.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.off-heap-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.off-heap-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.off-heap-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.owners

Number of cluster-wide replicas for each cache entry.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.partition-handling-component.availability

Indicates the current availability of the cache.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.partition-handling-component.enabled

If enabled, the cache will enter degraded mode upon detecting a network partition that threatens the integrity of the cache.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.passivations

The number of cache node passivations (passivating a node from memory to a cache store).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.queue-flush-interval

In ASYNC mode, this attribute controls how often the asynchronous thread used to flush the replication queue runs. This should be a positive integer which represents thread wakeup time in milliseconds.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.queue-size

In ASYNC mode, this attribute can be used to trigger flushing of the queue when it reaches a specific threshold.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.read-write-ratio

The read/write ratio of the cache ((hits+misses)/stores).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.cache

The name of the remote cache to use for this remote store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.remote-servers

A list of remote servers for this cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.socket-timeout

A socket timeout for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-store.tcp-no-delay

A TCP_NODELAY value for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remote-timeout

In SYNC mode, the timeout (in ms) used to wait for an acknowledgment when making a remote call, after which the call is aborted and an exception is thrown.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remove-hits

The number of cache attribute remove hits.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.remove-misses

The number of cache attribute remove misses.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.replication-count

The number of times data was replicated around the cluster.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.replication-failures

The number of data replication failures.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.segments

Controls the number of hash space segments which is the granularity for key distribution in the cluster. Value must be strictly positive.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.start

The cache start mode, which can be EAGER (immediate start) or LAZY (on-demand start).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.state-transfer-component.chunk-size

The maximum number of cache entries in a batch of transferred state.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.state-transfer-component.enabled

If enabled, this will cause the cache to ask neighboring caches for state when it starts up, so the cache starts 'warm', although it will impact startup time.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.state-transfer-component.timeout

The maximum amount of time (ms) to wait for state from neighboring caches, before throwing an exception and aborting startup. If timeout is 0, state transfer is performed asynchronously, and the cache will be immediately available.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.statistics-enabled

If enabled, statistics will be collected for this cache

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.stores

The number of cache attribute put operations.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.success-ratio

The data replication success ratio (successes/successes+failures).

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.time-since-reset

Time (in secs) since cache statistics were reset.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.transaction-component.commits

The number of transaction commits.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.transaction-component.locking

The locking mode for this cache, one of OPTIMISTIC or PESSIMISTIC.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.transaction-component.mode

Sets the cache transaction mode to one of NONE, NON_XA, NON_DURABLE_XA, FULL_XA.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.transaction-component.prepares

The number of transaction prepares.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.transaction-component.rollbacks

The number of transaction rollbacks.

thorntail.infinispan.cache-containers.*KEY*.distributed-caches.*KEY*.transaction-component.stop-timeout

If there are any ongoing transactions when a cache is stopped, Infinispan waits for ongoing remote and local transactions to finish. The amount of time to wait for is defined by the cache stop timeout.

thorntail.infinispan.cache-containers.*KEY*.eviction-executor

The scheduled executor used for eviction

thorntail.infinispan.cache-containers.*KEY*.expiration-thread-pool.keepalive-time

Used to specify the amount of milliseconds that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

thorntail.infinispan.cache-containers.*KEY*.expiration-thread-pool.max-threads

The maximum thread pool size.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.activations

The number of cache node activations (bringing a node into memory from a cache store).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.async-marshalling

If enabled, this will cause marshalling of entries to be performed asynchronously.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.average-read-time

Average time (in ms) for cache reads. Includes hits and misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.average-replication-time

The average time taken to replicate data around the cluster.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.average-write-time

Average time (in ms) for cache writes.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.batching

If enabled, the invocation batching API will be made available for this cache.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.binary-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.cache-status

The status of the cache component.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.attribute-class

The custom store implementation class to use for this cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.custom-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.elapsed-time

Time (in secs) since cache started.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.expiration-component.interval

Interval (in milliseconds) between subsequent runs to purge expired entries from memory and any cache stores. If you wish to disable the periodic eviction process altogether, set wakeupInterval to -1.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.expiration-component.lifespan

Maximum lifespan of a cache entry, after which the entry is expired cluster-wide, in milliseconds. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.expiration-component.max-idle

Maximum idle time a cache entry will be maintained in the cache, in milliseconds. If the idle time is exceeded, the entry will be expired cluster-wide. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.path

The system path under which this cache store will persist its entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.relative-to

The system path to which the specified path is relative.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.file-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hit-ratio

The hit/miss ratio for the cache (hits/hits+misses).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hits

The number of cache attribute hits.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.cache-configuration

Name of the cache configuration template defined in Infinispan Server to create caches from.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.remote-cache-container

Reference to a container-managed remote-cache-container.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.hotrod-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.indexing

If enabled, entries will be indexed when they are added to the cache. Indexes will be updated as entries change or are removed.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.indexing-properties

Properties to control indexing behaviour

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.invalidations

The number of cache invalidations.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.jndi-name

The jndi-name to which to bind this cache instance.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.locking-component.acquire-timeout

Maximum time to attempt a particular lock acquisition.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.locking-component.concurrency-level

Concurrency level for lock containers. Adjust this value according to the number of concurrent threads interacting with Infinispan.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.locking-component.current-concurrency-level

The estimated number of concurrently updating threads which this cache can support.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.locking-component.isolation

Sets the cache locking isolation level.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.locking-component.number-of-locks-available

The number of locks available to this cache.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.locking-component.number-of-locks-held

The number of locks currently in use by this cache.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.locking-component.striping

If true, a pool of shared locks is maintained for all entries that need to be locked. Otherwise, a lock is created per entry in the cache. Lock striping helps control memory footprint but may reduce concurrency in the system.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.misses

The number of cache attribute misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mixed-jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.mode

Sets the clustered cache mode, ASYNC for asynchronous operation, or SYNC for synchronous operation.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.module

The module whose class loader should be used when building this cache's configuration.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.number-of-entries

The number of entries in the cache including passivated entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.object-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.object-memory.max-entries

Maximum number of entries in a cache instance. If selected value is not a power of two the actual value will default to the least power of two larger than selected value. -1 means no limit.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.object-memory.size

Triggers eviction of the least recently used entries when the number of cache entries exceeds this threshold.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.object-memory.strategy

Sets the cache eviction strategy. Available options are 'UNORDERED', 'FIFO', 'LRU', 'LIRS' and 'NONE' (to disable eviction).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.off-heap-memory.capacity

Defines the capacity of the off-heap storage.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.off-heap-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.off-heap-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.off-heap-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.passivations

The number of cache node passivations (passivating a node from memory to a cache store).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.queue-flush-interval

In ASYNC mode, this attribute controls how often the asynchronous thread used to flush the replication queue runs. This should be a positive integer which represents thread wakeup time in milliseconds.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.queue-size

In ASYNC mode, this attribute can be used to trigger flushing of the queue when it reaches a specific threshold.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.read-write-ratio

The read/write ratio of the cache ((hits+misses)/stores).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.cache

The name of the remote cache to use for this remote store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.remote-servers

A list of remote servers for this cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.socket-timeout

A socket timeout for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-store.tcp-no-delay

A TCP_NODELAY value for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remote-timeout

In SYNC mode, the timeout (in ms) used to wait for an acknowledgment when making a remote call, after which the call is aborted and an exception is thrown.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remove-hits

The number of cache attribute remove hits.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.remove-misses

The number of cache attribute remove misses.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.replication-count

The number of times data was replicated around the cluster.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.replication-failures

The number of data replication failures.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.start

The cache start mode, which can be EAGER (immediate start) or LAZY (on-demand start).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.statistics-enabled

If enabled, statistics will be collected for this cache

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.stores

The number of cache attribute put operations.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.success-ratio

The data replication success ratio (successes/successes+failures).

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.time-since-reset

Time (in secs) since cache statistics were reset.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.transaction-component.commits

The number of transaction commits.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.transaction-component.locking

The locking mode for this cache, one of OPTIMISTIC or PESSIMISTIC.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.transaction-component.mode

Sets the cache transaction mode to one of NONE, NON_XA, NON_DURABLE_XA, FULL_XA.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.transaction-component.prepares

The number of transaction prepares.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.transaction-component.rollbacks

The number of transaction rollbacks.

thorntail.infinispan.cache-containers.*KEY*.invalidation-caches.*KEY*.transaction-component.stop-timeout

If there are any ongoing transactions when a cache is stopped, Infinispan waits for ongoing remote and local transactions to finish. The amount of time to wait for is defined by the cache stop timeout.

thorntail.infinispan.cache-containers.*KEY*.is-coordinator

Set to true if this node is the cluster's coordinator. May return null if the cache manager is not started.

thorntail.infinispan.cache-containers.*KEY*.jgroups-transport.channel

The channel of this cache container's transport.

thorntail.infinispan.cache-containers.*KEY*.jgroups-transport.cluster

The name of the group communication cluster

thorntail.infinispan.cache-containers.*KEY*.jgroups-transport.executor

The executor to use for the transport

thorntail.infinispan.cache-containers.*KEY*.jgroups-transport.lock-timeout

The timeout for locks for the transport

thorntail.infinispan.cache-containers.*KEY*.jgroups-transport.stack

The jgroups stack to use for the transport

thorntail.infinispan.cache-containers.*KEY*.jndi-name

The jndi name to which to bind this cache container

thorntail.infinispan.cache-containers.*KEY*.listener-executor

The executor used for the replication queue

thorntail.infinispan.cache-containers.*KEY*.listener-thread-pool.keepalive-time

Used to specify the amount of milliseconds that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

thorntail.infinispan.cache-containers.*KEY*.listener-thread-pool.max-threads

The maximum thread pool size.

thorntail.infinispan.cache-containers.*KEY*.listener-thread-pool.min-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.

thorntail.infinispan.cache-containers.*KEY*.listener-thread-pool.queue-length

The queue length.

thorntail.infinispan.cache-containers.*KEY*.local-address

The local address of the node. May return null if the cache manager is not started.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.activations

The number of cache node activations (bringing a node into memory from a cache store).

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.average-read-time

Average time (in ms) for cache reads. Includes hits and misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.average-write-time

Average time (in ms) for cache writes.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.batching

If enabled, the invocation batching API will be made available for this cache.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.binary-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.cache-status

The status of the cache component.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.attribute-class

The custom store implementation class to use for this cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.custom-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.elapsed-time

Time (in secs) since cache started.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.expiration-component.interval

Interval (in milliseconds) between subsequent runs to purge expired entries from memory and any cache stores. If you wish to disable the periodic eviction process altogether, set wakeupInterval to -1.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.expiration-component.lifespan

Maximum lifespan of a cache entry, after which the entry is expired cluster-wide, in milliseconds. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.expiration-component.max-idle

Maximum idle time a cache entry will be maintained in the cache, in milliseconds. If the idle time is exceeded, the entry will be expired cluster-wide. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.path

The system path under which this cache store will persist its entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.relative-to

The system path to which the specified path is relative.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.file-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hit-ratio

The hit/miss ratio for the cache (hits/hits+misses).

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hits

The number of cache attribute hits.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.cache-configuration

Name of the cache configuration template defined in Infinispan Server to create caches from.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.remote-cache-container

Reference to a container-managed remote-cache-container.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.hotrod-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.indexing

If enabled, entries will be indexed when they are added to the cache. Indexes will be updated as entries change or are removed.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.indexing-properties

Properties to control indexing behaviour

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.invalidations

The number of cache invalidations.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.jndi-name

The jndi-name to which to bind this cache instance.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.locking-component.acquire-timeout

Maximum time to attempt a particular lock acquisition.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.locking-component.concurrency-level

Concurrency level for lock containers. Adjust this value according to the number of concurrent threads interacting with Infinispan.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.locking-component.current-concurrency-level

The estimated number of concurrently updating threads which this cache can support.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.locking-component.isolation

Sets the cache locking isolation level.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.locking-component.number-of-locks-available

The number of locks available to this cache.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.locking-component.number-of-locks-held

The number of locks currently in use by this cache.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.locking-component.striping

If true, a pool of shared locks is maintained for all entries that need to be locked. Otherwise, a lock is created per entry in the cache. Lock striping helps control memory footprint but may reduce concurrency in the system.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.misses

The number of cache attribute misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.mixed-jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.module

The module whose class loader should be used when building this cache’s configuration.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.number-of-entries

The number of entries in the cache including passivated entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.object-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.object-memory.max-entries

Maximum number of entries in a cache instance. If selected value is not a power of two the actual value will default to the least power of two larger than selected value. -1 means no limit.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.object-memory.size

Triggers eviction of the least recently used entries when the number of cache entries exceeds this threshold.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.object-memory.strategy

Sets the cache eviction strategy. Available options are 'UNORDERED', 'FIFO', 'LRU', 'LIRS' and 'NONE' (to disable eviction).

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.off-heap-memory.capacity

Defines the capacity of the off-heap storage.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.off-heap-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.off-heap-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.off-heap-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.passivations

The number of cache node passivations (passivating a node from memory to a cache store).

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.read-write-ratio

The read/write ratio of the cache ((hits+misses)/stores).

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.cache

The name of the remote cache to use for this remote store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.remote-servers

A list of remote servers for this cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.socket-timeout

A socket timeout for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remote-store.tcp-no-delay

A TCP_NODELAY value for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remove-hits

The number of cache attribute remove hits.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.remove-misses

The number of cache attribute remove misses.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.start

The cache start mode, which can be EAGER (immediate start) or LAZY (on-demand start).

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.statistics-enabled

If enabled, statistics will be collected for this cache

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.stores

The number of cache attribute put operations.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.time-since-reset

Time (in secs) since cache statistics were reset.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.transaction-component.commits

The number of transaction commits.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.transaction-component.locking

The locking mode for this cache, one of OPTIMISTIC or PESSIMISTIC.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.transaction-component.mode

Sets the cache transaction mode to one of NONE, NON_XA, NON_DURABLE_XA, FULL_XA.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.transaction-component.prepares

The number of transaction prepares.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.transaction-component.rollbacks

The number of transaction rollbacks.

thorntail.infinispan.cache-containers.*KEY*.local-caches.*KEY*.transaction-component.stop-timeout

If there are any ongoing transactions when a cache is stopped, Infinispan waits for ongoing remote and local transactions to finish. The amount of time to wait for is defined by the cache stop timeout.

thorntail.infinispan.cache-containers.*KEY*.module

The module whose class loader should be used when building this cache container's configuration.

thorntail.infinispan.cache-containers.*KEY*.persistence-thread-pool.keepalive-time

Used to specify the amount of milliseconds that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

thorntail.infinispan.cache-containers.*KEY*.persistence-thread-pool.max-threads

The maximum thread pool size.

thorntail.infinispan.cache-containers.*KEY*.persistence-thread-pool.min-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.

thorntail.infinispan.cache-containers.*KEY*.persistence-thread-pool.queue-length

The queue length.

thorntail.infinispan.cache-containers.*KEY*.remote-command-thread-pool.keepalive-time

Used to specify the amount of milliseconds that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

thorntail.infinispan.cache-containers.*KEY*.remote-command-thread-pool.max-threads

The maximum thread pool size.

thorntail.infinispan.cache-containers.*KEY*.remote-command-thread-pool.min-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.

thorntail.infinispan.cache-containers.*KEY*.remote-command-thread-pool.queue-length

The queue length.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.activations

The number of cache node activations (bringing a node into memory from a cache store).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.async-marshalling

If enabled, this will cause marshalling of entries to be performed asynchronously.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.average-read-time

Average time (in ms) for cache reads. Includes hits and misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.average-replication-time

The average time taken to replicate data around the cluster.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.average-write-time

Average time (in ms) for cache writes.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.backup-for-component.remote-cache

The name of the remote cache for which this cache acts as a backup.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.backup-for-component.remote-site

The site of the remote cache for which this cache acts as a backup.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.backups-component.backups.*KEY*.after-failures

Indicates the number of failures after which this backup site should go offline.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.backups-component.backups.*KEY*.enabled

Indicates whether or not this backup site is enabled.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.backups-component.backups.*KEY*.failure-policy

The policy to follow when connectivity to the backup site fails.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.backups-component.backups.*KEY*.min-wait

Indicates the minimum time (in milliseconds) to wait after the max number of failures is reached, after which this backup site should go offline.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.backups-component.backups.*KEY*.strategy

The backup strategy for this cache

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.backups-component.backups.*KEY*.timeout

The timeout for replicating to the backup site.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.batching

If enabled, the invocation batching API will be made available for this cache.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.binary-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.cache-status

The status of the cache component.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.attribute-class

The custom store implementation class to use for this cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.custom-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.elapsed-time

Time (in secs) since cache started.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.expiration-component.interval

Interval (in milliseconds) between subsequent runs to purge expired entries from memory and any cache stores. If you wish to disable the periodic eviction process altogether, set wakeupInterval to -1.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.expiration-component.lifespan

Maximum lifespan of a cache entry, after which the entry is expired cluster-wide, in milliseconds. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.expiration-component.max-idle

Maximum idle time a cache entry will be maintained in the cache, in milliseconds. If the idle time is exceeded, the entry will be expired cluster-wide. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.path

The system path under which this cache store will persist its entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.relative-to

The system path to which the specified path is relative.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.file-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hit-ratio

The hit/miss ratio for the cache (hits/hits+misses).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hits

The number of cache attribute hits.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.cache-configuration

Name of the cache configuration template defined in Infinispan Server to create caches from.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.remote-cache-container

Reference to a container-managed remote-cache-container.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.hotrod-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.indexing

If enabled, entries will be indexed when they are added to the cache. Indexes will be updated as entries change or are removed.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.indexing-properties

Properties to control indexing behaviour

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.invalidations

The number of cache invalidations.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.jndi-name

The jndi-name to which to bind this cache instance.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.locking-component.acquire-timeout

Maximum time to attempt a particular lock acquisition.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.locking-component.concurrency-level

Concurrency level for lock containers. Adjust this value according to the number of concurrent threads interacting with Infinispan.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.locking-component.current-concurrency-level

The estimated number of concurrently updating threads which this cache can support.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.locking-component.isolation

Sets the cache locking isolation level.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.locking-component.number-of-locks-available

The number of locks available to this cache.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.locking-component.number-of-locks-held

The number of locks currently in use by this cache.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.locking-component.striping

If true, a pool of shared locks is maintained for all entries that need to be locked. Otherwise, a lock is created per entry in the cache. Lock striping helps control memory footprint but may reduce concurrency in the system.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.misses

The number of cache attribute misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mixed-jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.mode

Sets the clustered cache mode, ASYNC for asynchronous operation, or SYNC for synchronous operation.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.module

The module whose class loader should be used when building this cache’s configuration.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.number-of-entries

The number of entries in the cache including passivated entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.object-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.object-memory.max-entries

Maximum number of entries in a cache instance. If selected value is not a power of two the actual value will default to the least power of two larger than selected value. -1 means no limit.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.object-memory.size

Triggers eviction of the least recently used entries when the number of cache entries exceeds this threshold.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.object-memory.strategy

Sets the cache eviction strategy. Available options are 'UNORDERED', 'FIFO', 'LRU', 'LIRS' and 'NONE' (to disable eviction).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.off-heap-memory.capacity

Defines the capacity of the off-heap storage.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.off-heap-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.off-heap-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.off-heap-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.partition-handling-component.availability

Indicates the current availability of the cache.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.partition-handling-component.enabled

If enabled, the cache will enter degraded mode upon detecting a network partition that threatens the integrity of the cache.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.passivations

The number of cache node passivations (passivating a node from memory to a cache store).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.queue-flush-interval

In ASYNC mode, this attribute controls how often the asynchronous thread used to flush the replication queue runs. This should be a positive integer which represents thread wakeup time in milliseconds.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.queue-size

In ASYNC mode, this attribute can be used to trigger flushing of the queue when it reaches a specific threshold.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.read-write-ratio

The read/write ratio of the cache ((hits+misses)/stores).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.cache

The name of the remote cache to use for this remote store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.remote-servers

A list of remote servers for this cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.socket-timeout

A socket timeout for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-store.tcp-no-delay

A TCP_NODELAY value for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remote-timeout

In SYNC mode, the timeout (in ms) used to wait for an acknowledgment when making a remote call, after which the call is aborted and an exception is thrown.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remove-hits

The number of cache attribute remove hits.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.remove-misses

The number of cache attribute remove misses.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.replication-count

The number of times data was replicated around the cluster.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.replication-failures

The number of data replication failures.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.start

The cache start mode, which can be EAGER (immediate start) or LAZY (on-demand start).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.state-transfer-component.chunk-size

The maximum number of cache entries in a batch of transferred state.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.state-transfer-component.enabled

If enabled, this will cause the cache to ask neighboring caches for state when it starts up, so the cache starts 'warm', although it will impact startup time.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.state-transfer-component.timeout

The maximum amount of time (ms) to wait for state from neighboring caches, before throwing an exception and aborting startup. If timeout is 0, state transfer is performed asynchronously, and the cache will be immediately available.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.statistics-enabled

If enabled, statistics will be collected for this cache

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.stores

The number of cache attribute put operations.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.success-ratio

The data replication success ratio (successes/successes+failures).

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.time-since-reset

Time (in secs) since cache statistics were reset.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.transaction-component.commits

The number of transaction commits.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.transaction-component.locking

The locking mode for this cache, one of OPTIMISTIC or PESSIMISTIC.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.transaction-component.mode

Sets the cache transaction mode to one of NONE, NON_XA, NON_DURABLE_XA, FULL_XA.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.transaction-component.prepares

The number of transaction prepares.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.transaction-component.rollbacks

The number of transaction rollbacks.

thorntail.infinispan.cache-containers.*KEY*.replicated-caches.*KEY*.transaction-component.stop-timeout

If there are any ongoing transactions when a cache is stopped, Infinispan waits for ongoing remote and local transactions to finish. The amount of time to wait for is defined by the cache stop timeout.

thorntail.infinispan.cache-containers.*KEY*.replication-queue-executor

The executor used for asynchronous cache operations

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.activations

The number of cache node activations (bringing a node into memory from a cache store).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.async-marshalling

If enabled, this will cause marshalling of entries to be performed asynchronously.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.average-read-time

Average time (in ms) for cache reads. Includes hits and misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.average-replication-time

The average time taken to replicate data around the cluster.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.average-write-time

Average time (in ms) for cache writes.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.backup-for-component.remote-cache

The name of the remote cache for which this cache acts as a backup.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.backup-for-component.remote-site

The site of the remote cache for which this cache acts as a backup.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.backups-component.backups.*KEY*.after-failures

Indicates the number of failures after which this backup site should go offline.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.backups-component.backups.*KEY*.enabled

Indicates whether or not this backup site is enabled.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.backups-component.backups.*KEY*.failure-policy

The policy to follow when connectivity to the backup site fails.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.backups-component.backups.*KEY*.min-wait

Indicates the minimum time (in milliseconds) to wait after the max number of failures is reached, after which this backup site should go offline.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.backups-component.backups.*KEY*.strategy

The backup strategy for this cache

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.backups-component.backups.*KEY*.timeout

The timeout for replicating to the backup site.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.batching

If enabled, the invocation batching API will be made available for this cache.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.bias-lifespan

When greater than zero, specifies the duration (in ms) that a cache entry will be cached on a non-owner following a write operation.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.binary-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.cache-status

The status of the cache component.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.consistent-hash-strategy

Defines the consistent hash strategy for the cache.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.attribute-class

The custom store implementation class to use for this cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.custom-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.elapsed-time

Time (in secs) since cache started.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.expiration-component.interval

Interval (in milliseconds) between subsequent runs to purge expired entries from memory and any cache stores. If you wish to disable the periodic eviction process altogether, set wakeupInterval to -1.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.expiration-component.lifespan

Maximum lifespan of a cache entry, after which the entry is expired cluster-wide, in milliseconds. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.expiration-component.max-idle

Maximum idle time a cache entry will be maintained in the cache, in milliseconds. If the idle time is exceeded, the entry will be expired cluster-wide. -1 means the entries never expire.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.path

The system path under which this cache store will persist its entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.relative-to

The system path to which the specified path is relative.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.file-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hit-ratio

The hit/miss ratio for the cache (hits/hits+misses).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hits

The number of cache attribute hits.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.cache-configuration

Name of the cache configuration template defined in Infinispan Server to create caches from.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.remote-cache-container

Reference to a container-managed remote-cache-container.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.hotrod-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.indexing

If enabled, entries will be indexed when they are added to the cache. Indexes will be updated as entries change or are removed.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.indexing-properties

Properties to control indexing behaviour

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.invalidation-batch-size

The threshold after which batched invalidations are sent.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.invalidations

The number of cache invalidations.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.jndi-name

The jndi-name to which to bind this cache instance.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.locking-component.acquire-timeout

Maximum time to attempt a particular lock acquisition.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.locking-component.concurrency-level

Concurrency level for lock containers. Adjust this value according to the number of concurrent threads interacting with Infinispan.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.locking-component.current-concurrency-level

The estimated number of concurrently updating threads which this cache can support.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.locking-component.isolation

Sets the cache locking isolation level.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.locking-component.number-of-locks-available

The number of locks available to this cache.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.locking-component.number-of-locks-held

The number of locks currently in use by this cache.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.locking-component.striping

If true, a pool of shared locks is maintained for all entries that need to be locked. Otherwise, a lock is created per entry in the cache. Lock striping helps control memory footprint but may reduce concurrency in the system.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.misses

The number of cache attribute misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-keyed-table

Defines a table used to store cache entries whose keys cannot be expressed as strings.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.binary-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.data-source

References the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.datasource

The jndi name of the data source used to connect to this store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.dialect

The dialect of this datastore.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-keyed-table

Defines a table used to store persistent cache entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-table.batch-size

For DB inserts, the batch size determines how many inserts are batched together.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-table.create-on-start

Indicates whether the store should create this database table when the cache starts.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-table.data-column

A database column to hold cache entry data.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-table.drop-on-stop

Indicates whether the store should drop this database table when the cache stops.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-table.fetch-size

For DB queries, the fetch size will be used to set the fetch size on ResultSets.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-table.id-column

A database column to hold cache entry ids.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-table.prefix

The prefix for the database table name.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mixed-jdbc-store.string-table.timestamp-column

A database column to hold cache entry timestamps.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.mode

Sets the clustered cache mode, ASYNC for asynchronous operation, or SYNC for synchronous operation.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.module

The module whose class loader should be used when building this cache’s configuration.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.number-of-entries

The number of entries in the cache including passivated entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.object-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.object-memory.max-entries

Maximum number of entries in a cache instance. If selected value is not a power of two the actual value will default to the least power of two larger than selected value. -1 means no limit.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.object-memory.size

Triggers eviction of the least recently used entries when the number of cache entries exceeds this threshold.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.object-memory.strategy

Sets the cache eviction strategy. Available options are 'UNORDERED', 'FIFO', 'LRU', 'LIRS' and 'NONE' (to disable eviction).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.off-heap-memory.capacity

Defines the capacity of the off-heap storage.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.off-heap-memory.eviction-type

Indicates whether the size attribute refers to the number of cache entries (i.e. COUNT) or the collective size of the cache entries (i.e. MEMORY).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.off-heap-memory.evictions

The number of cache eviction operations.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.off-heap-memory.size

Eviction threshold, as defined by the eviction-type.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.partition-handling-component.availability

Indicates the current availability of the cache.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.partition-handling-component.enabled

If enabled, the cache will enter degraded mode upon detecting a network partition that threatens the integrity of the cache.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.passivations

The number of cache node passivations (passivating a node from memory to a cache store).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.queue-flush-interval

In ASYNC mode, this attribute controls how often the asynchronous thread used to flush the replication queue runs. This should be a positive integer which represents thread wakeup time in milliseconds.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.queue-size

In ASYNC mode, this attribute can be used to trigger flushing of the queue when it reaches a specific threshold.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.read-write-ratio

The read/write ratio of the cache ((hits+misses)/stores).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.behind-write.flush-lock-timeout

Timeout to acquire the lock which guards the state to be flushed to the cache store periodically.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.behind-write.modification-queue-size

Maximum number of entries in the asynchronous queue. When the queue is full, the store becomes write-through until it can accept new entries.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.behind-write.shutdown-timeout

Timeout in milliseconds to stop the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.behind-write.thread-pool-size

Size of the thread pool whose threads are responsible for applying the modifications to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.cache

The name of the remote cache to use for this remote store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.cache-loader-loads

The number of cache loader node loads.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.cache-loader-misses

The number of cache loader node misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.fetch-state

If true, fetch persistent state when joining a cluster. If multiple cache stores are chained, only one of them can have this property enabled.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.max-batch-size

The maximum size of a batch to be inserted/deleted from the store. If the value is less than one, then no upper limit is placed on the number of operations in a batch.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.passivation

If true, data is only written to the cache store when it is evicted from memory, a phenomenon known as 'passivation'. Next time the data is requested, it will be 'activated' which means that data will be brought back to memory and removed from the persistent store. If false, the cache store contains a copy of the contents in memory, so writes to cache result in cache store writes. This essentially gives you a 'write-through' configuration.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.preload

If true, when the cache starts, data stored in the cache store will be pre-loaded into memory. This is particularly useful when data in the cache store will be needed immediately after startup and you want to avoid cache operations being delayed as a result of loading this data lazily. Can be used to provide a 'warm-cache' on startup, however there is a performance penalty as startup time is affected by this process.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.properties

A list of cache store properties.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.properties.*KEY*.value

The value of the cache store property.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.purge

If true, purges this cache store when it starts up.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.remote-servers

A list of remote servers for this cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.shared

This setting should be set to true when multiple cache instances share the same cache store (e.g., multiple nodes in a cluster using a JDBC-based CacheStore pointing to the same, shared database.) Setting this to true avoids multiple cache instances writing the same modification multiple times. If enabled, only the node where the modification originated will write to the cache store. If disabled, each individual cache reacts to a potential remote update by storing the data to the cache store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.singleton

If true, the singleton store cache store is enabled. SingletonStore is a delegating cache store used for situations when only one instance in a cluster should interact with the underlying store.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.socket-timeout

A socket timeout for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-store.tcp-no-delay

A TCP_NODELAY value for remote cache communication.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remote-timeout

In SYNC mode, the timeout (in ms) used to wait for an acknowledgment when making a remote call, after which the call is aborted and an exception is thrown.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remove-hits

The number of cache attribute remove hits.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.remove-misses

The number of cache attribute remove misses.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.replication-count

The number of times data was replicated around the cluster.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.replication-failures

The number of data replication failures.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.segments

Controls the number of hash space segments which is the granularity for key distribution in the cluster. Value must be strictly positive.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.start

The cache start mode, which can be EAGER (immediate start) or LAZY (on-demand start).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.state-transfer-component.chunk-size

The maximum number of cache entries in a batch of transferred state.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.state-transfer-component.enabled

If enabled, this will cause the cache to ask neighboring caches for state when it starts up, so the cache starts 'warm', although it will impact startup time.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.state-transfer-component.timeout

The maximum amount of time (ms) to wait for state from neighboring caches, before throwing an exception and aborting startup. If timeout is 0, state transfer is performed asynchronously, and the cache will be immediately available.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.statistics-enabled

If enabled, statistics will be collected for this cache

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.stores

The number of cache attribute put operations.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.success-ratio

The data replication success ratio (successes/successes+failures).

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.time-since-reset

Time (in secs) since cache statistics were reset.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.transaction-component.commits

The number of transaction commits.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.transaction-component.locking

The locking mode for this cache, one of OPTIMISTIC or PESSIMISTIC.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.transaction-component.mode

Sets the cache transaction mode to one of NONE, NON_XA, NON_DURABLE_XA, FULL_XA.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.transaction-component.prepares

The number of transaction prepares.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.transaction-component.rollbacks

The number of transaction rollbacks.

thorntail.infinispan.cache-containers.*KEY*.scattered-caches.*KEY*.transaction-component.stop-timeout

If there are any ongoing transactions when a cache is stopped, Infinispan waits for ongoing remote and local transactions to finish. The amount of time to wait for is defined by the cache stop timeout.

thorntail.infinispan.cache-containers.*KEY*.start

The cache container start mode, which can be EAGER (immediate start) or LAZY (on-demand start).

thorntail.infinispan.cache-containers.*KEY*.state-transfer-thread-pool.keepalive-time

Used to specify the amount of milliseconds that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

thorntail.infinispan.cache-containers.*KEY*.state-transfer-thread-pool.max-threads

The maximum thread pool size.

thorntail.infinispan.cache-containers.*KEY*.state-transfer-thread-pool.min-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.

thorntail.infinispan.cache-containers.*KEY*.state-transfer-thread-pool.queue-length

The queue length.

thorntail.infinispan.cache-containers.*KEY*.statistics-enabled

If enabled, statistics will be collected for this cache container

thorntail.infinispan.cache-containers.*KEY*.transport-thread-pool.keepalive-time

Used to specify the amount of milliseconds that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

thorntail.infinispan.cache-containers.*KEY*.transport-thread-pool.max-threads

The maximum thread pool size.

thorntail.infinispan.cache-containers.*KEY*.transport-thread-pool.min-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.

thorntail.infinispan.cache-containers.*KEY*.transport-thread-pool.queue-length

The queue length.

thorntail.infinispan.default-fraction

(not yet documented)

thorntail.infinispan.remote-cache-containers.*KEY*.active-connections

The number of active connections to the Infinispan server.

thorntail.infinispan.remote-cache-containers.*KEY*.async-thread-pool.keepalive-time

Used to specify the amount of milliseconds that pool threads should be kept running when idle; if not specified, threads will run until the executor is shut down.

thorntail.infinispan.remote-cache-containers.*KEY*.async-thread-pool.max-threads

The maximum thread pool size.

thorntail.infinispan.remote-cache-containers.*KEY*.async-thread-pool.min-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.

thorntail.infinispan.remote-cache-containers.*KEY*.async-thread-pool.queue-length

The queue length.

thorntail.infinispan.remote-cache-containers.*KEY*.connection-pool-component.exhausted-action

Specifies what happens when asking for a connection from a server's pool, and that pool is exhausted.

thorntail.infinispan.remote-cache-containers.*KEY*.connection-pool-component.max-active

Controls the maximum number of connections per server that are allocated (checked out to client threads, or idle in the pool) at one time. When non-positive, there is no limit to the number of connections per server. When maxActive is reached, the connection pool for that server is said to be exhausted. Value -1 means no limit.

thorntail.infinispan.remote-cache-containers.*KEY*.connection-pool-component.max-wait

The amount of time in milliseconds to wait for a connection to become available when the exhausted action is ExhaustedAction.WAIT, after which a java.util.NoSuchElementException will be thrown. If a negative value is supplied, the pool will block indefinitely.

thorntail.infinispan.remote-cache-containers.*KEY*.connection-pool-component.min-evictable-idle-time

Specifies the minimum amount of time that an connection may sit idle in the pool before it is eligible for eviction due to idle time. When non-positive, no connection will be dropped from the pool due to idle time alone. This setting has no effect unless timeBetweenEvictionRunsMillis > 0.

thorntail.infinispan.remote-cache-containers.*KEY*.connection-pool-component.min-idle

Sets a target value for the minimum number of idle connections (per server) that should always be available. If this parameter is set to a positive number and timeBetweenEvictionRunsMillis > 0, each time the idle connection eviction thread runs, it will try to create enough idle instances so that there will be minIdle idle instances available for each server.

thorntail.infinispan.remote-cache-containers.*KEY*.connection-timeout

Defines the maximum socket connect timeout before giving up connecting to the server.

thorntail.infinispan.remote-cache-containers.*KEY*.connections

The total number of connections to the Infinispan server.

thorntail.infinispan.remote-cache-containers.*KEY*.default-remote-cluster

Required default remote server cluster.

thorntail.infinispan.remote-cache-containers.*KEY*.idle-connections

The number of idle connections to the Infinispan server.

thorntail.infinispan.remote-cache-containers.*KEY*.invalidation-near-cache.max-entries

Defines the maximum number of elements to keep in the near cache.

thorntail.infinispan.remote-cache-containers.*KEY*.key-size-estimate

This hint allows sizing of byte buffers when serializing and deserializing keys, to minimize array resizing.

thorntail.infinispan.remote-cache-containers.*KEY*.max-retries

Sets the maximum number of retries for each request. A valid value should be greater or equals than 0. Zero means no retry will made in case of a network failure.

thorntail.infinispan.remote-cache-containers.*KEY*.module

Defines the module whose class loader should be used when configuring remote cache container marshaller.

thorntail.infinispan.remote-cache-containers.*KEY*.properties

A list of remote cache container properties.

thorntail.infinispan.remote-cache-containers.*KEY*.protocol-version

This property defines the protocol version that this client should use.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.average-read-time

The average read time, in milliseconds, for this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.average-remove-time

The average remove time, in milliseconds, for this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.average-write-time

The average write time, in milliseconds, to this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.hits

The number of hits to this remote cache, excluding hits from the near-cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.misses

The number of misses to this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.near-cache-hits

The number of near-cache hits for this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.near-cache-invalidations

The number of near-cache invalidations for this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.near-cache-misses

The number of near-cache misses for this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.near-cache-size

The number of entries in the near-cache for this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.removes

The number of removes to this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.time-since-reset

The number of seconds since statistics were reset on this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-caches.*KEY*.writes

The number of writes to this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.remote-clusters.*KEY*.socket-bindings

List of outbound-socket-bindings of Hot Rod servers to connect to.

thorntail.infinispan.remote-cache-containers.*KEY*.security-component.ssl-context

Reference to the Elytron-managed SSLContext to be used for connecting to the remote cluster.

thorntail.infinispan.remote-cache-containers.*KEY*.socket-timeout

Enable or disable SO_TIMEOUT on socket connections to remote Hot Rod servers with the specified timeout, in milliseconds. A timeout of 0 is interpreted as an infinite timeout.

thorntail.infinispan.remote-cache-containers.*KEY*.statistics-enabled

Enables statistics gathering for this remote cache.

thorntail.infinispan.remote-cache-containers.*KEY*.tcp-keep-alive

Configures TCP Keepalive on the TCP stack.

thorntail.infinispan.remote-cache-containers.*KEY*.tcp-no-delay

Enable or disable TCP_NODELAY on socket connections to remote Hot Rod servers.

thorntail.infinispan.remote-cache-containers.*KEY*.transaction-component.mode

Sets the cache transaction mode to one of NONE, NON_XA, NON_DURABLE_XA, FULL_XA.

thorntail.infinispan.remote-cache-containers.*KEY*.transaction-component.timeout

The duration (in ms) after which idle transactions are rolled back.

thorntail.infinispan.remote-cache-containers.*KEY*.value-size-estimate

This hint allows sizing of byte buffers when serializing and deserializing values, to minimize array resizing.

D.14. 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

thorntail.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.

thorntail.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.

thorntail.io.buffer-pools.*KEY*.direct-buffers

Does the buffer pool use direct buffers, some platforms don’t support direct buffers

thorntail.io.workers.*KEY*.busy-task-thread-count

An estimate of busy threads in the task worker thread pool

thorntail.io.workers.*KEY*.core-pool-size

Minimum number of threads to keep in the underlying thread pool even if they are idle. Threads over this limit will be terminated over time specified by task-keepalive attribute.

thorntail.io.workers.*KEY*.io-thread-count

I/O thread count

thorntail.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

thorntail.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.

thorntail.io.workers.*KEY*.outbound-bind-address.*KEY*.bind-address

The address to bind to when the destination address matches

thorntail.io.workers.*KEY*.outbound-bind-address.*KEY*.bind-port

The port number to bind to when the destination address matches

thorntail.io.workers.*KEY*.outbound-bind-address.*KEY*.match

The destination address range to match

thorntail.io.workers.*KEY*.queue-size

An estimate of the number of tasks in the worker queue.

thorntail.io.workers.*KEY*.servers.*KEY*.connection-count

Estimate of the current connection count

thorntail.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.

thorntail.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.

thorntail.io.workers.*KEY*.shutdown-requested

True is shutdown of the pool was requested

thorntail.io.workers.*KEY*.stack-size

The stack size (in bytes) to attempt to use for worker threads.

thorntail.io.workers.*KEY*.task-core-threads

Specify the starting number of threads for the worker task thread pool.

thorntail.io.workers.*KEY*.task-keepalive

Specify the number of milliseconds to keep non-core task threads alive.

thorntail.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.15. JAEGER

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaeger</artifactId>
</dependency>
```

Configuration

thorntail.jaeger.agent-host

The hostname for communicating with agent via UDP

thorntail.jaeger.agent-port

The port for communicating with agent via UDP

thorntail.jaeger.enable-b3-header-propagation

Whether to enable propagation of B3 headers in the configured Tracer. By default this is false.

thorntail.jaeger.password

Password to send as part of "Basic" authentication to the endpoint

thorntail.jaeger.remote-reporter-http-endpoint

Remote Reporter HTTP endpoint for Jaeger collector, such as http://jaeger-collector.istio-system:14268/api/traces

thorntail.jaeger.reporter-flush-interval

The reporter’s flush interval (ms)

thorntail.jaeger.reporter-log-spans

Whether the reporter should also log the spans

thorntail.jaeger.reporter-max-queue-size

The reporter’s maximum queue size

thorntail.jaeger.sampler-manager-host

The host name and port when using the remote controlled sampler

thorntail.jaeger.sampler-parameter

The sampler parameter (number). Ex.: **1**

thorntail.jaeger.sampler-type

The sampler type. Ex.: **const**

thorntail.jaeger.service-name

The service name. Required (via this parameter, system property or env var). Ex.: **order-manager**

thorntail.jaeger.user

Username to send as part of "Basic" authentication to the endpoint

D.16. 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

thorntail.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.16.1. 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.16.2. 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.16.3. JAX-RS + JSON-B

Provides support within JAX-RS application for JSON Binding according to JSR-367.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaxrs-jsonb</artifactId>
</dependency>
```

D.16.4. 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.16.5. 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.16.6. 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.17. 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

thorntail.jca.archive-validation.enabled

Specify whether archive validation is enabled

thorntail.jca.archive-validation.fail-on-error

Should an archive validation error report fail the deployment

thorntail.jca.archive-validation.fail-on-warn

Should an archive validation warning report fail the deployment

thorntail.jca.bean-validation.enabled

Specify whether bean validation is enabled

thorntail.jca.bootstrap-contexts.*KEY*.name

The name of the BootstrapContext

thorntail.jca.bootstrap-contexts.*KEY*.workmanager

The WorkManager instance for the BootstrapContext

thorntail.jca.cached-connection-manager.debug

Enable/disable debug information logging

thorntail.jca.cached-connection-manager.error

Enable/disable error information logging

thorntail.jca.cached-connection-manager.ignore-unknown-connections

Do not cache unknown connections

thorntail.jca.cached-connection-manager.install

Enable/disable the cached connection manager valve and interceptor

thorntail.jca.distributed-workmanagers.*KEY*.elytron-enabled

Enables Elytron security for this workmanager.

thorntail.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.allow-core-timeout

Whether core threads may time out.

thorntail.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.

thorntail.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.current-thread-count

The current number of threads in the pool.

thorntail.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.

thorntail.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

thorntail.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.max-threads

The maximum thread pool size.

thorntail.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.name

The name of the thread pool.

thorntail.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.queue-length

The queue length.

thorntail.jca.distributed-workmanagers.*KEY*.long-running-threads.*KEY*.queue-size

The queue size.

thorntail.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.

thorntail.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.

thorntail.jca.distributed-workmanagers.*KEY*.name

The name of the DistributedWorkManager

thorntail.jca.distributed-workmanagers.*KEY*.policy

The policy decides when to redistribute a Work instance

thorntail.jca.distributed-workmanagers.*KEY*.policy-options

List of policy’s options key/value pairs

thorntail.jca.distributed-workmanagers.*KEY*.selector

The selector decides to which nodes in the network to redistribute the Work instance to

thorntail.jca.distributed-workmanagers.*KEY*.selector-options

List of selector’s options key/value pairs

thorntail.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.allow-core-timeout

Whether core threads may time out.

thorntail.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.

thorntail.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.current-thread-count

The current number of threads in the pool.

thorntail.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.

thorntail.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

thorntail.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.max-threads

The maximum thread pool size.

thorntail.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.name

The name of the thread pool.

thorntail.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.queue-length

The queue length.

thorntail.jca.distributed-workmanagers.*KEY*.short-running-threads.*KEY*.queue-size

The queue size.

thorntail.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.

thorntail.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.

thorntail.jca.tracer.enabled

Specify whether tracer is enabled

thorntail.jca.workmanagers.*KEY*.elytron-enabled

Enables Elytron security for this workmanager.

thorntail.jca.workmanagers.*KEY*.long-running-threads.*KEY*.allow-core-timeout

Whether core threads may time out.

thorntail.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.

thorntail.jca.workmanagers.*KEY*.long-running-threads.*KEY*.current-thread-count

The current number of threads in the pool.

thorntail.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.

thorntail.jca.workmanagers.*KEY*.long-running-threads.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

thorntail.jca.workmanagers.*KEY*.long-running-threads.*KEY*.max-threads

The maximum thread pool size.

thorntail.jca.workmanagers.*KEY*.long-running-threads.*KEY*.name

The name of the thread pool.

thorntail.jca.workmanagers.*KEY*.long-running-threads.*KEY*.queue-length

The queue length.

thorntail.jca.workmanagers.*KEY*.long-running-threads.*KEY*.queue-size

The queue size.

thorntail.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.

thorntail.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.

thorntail.jca.workmanagers.*KEY*.name

The name of the WorkManager

thorntail.jca.workmanagers.*KEY*.short-running-threads.*KEY*.allow-core-timeout

Whether core threads may time out.

thorntail.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.

thorntail.jca.workmanagers.*KEY*.short-running-threads.*KEY*.current-thread-count

The current number of threads in the pool.

thorntail.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.

thorntail.jca.workmanagers.*KEY*.short-running-threads.*KEY*.largest-thread-count

The largest number of threads that have ever simultaneously been in the pool.

thorntail.jca.workmanagers.*KEY*.short-running-threads.*KEY*.max-threads

The maximum thread pool size.

thorntail.jca.workmanagers.*KEY*.short-running-threads.*KEY*.name

The name of the thread pool.

thorntail.jca.workmanagers.*KEY*.short-running-threads.*KEY*.queue-length

The queue length.

thorntail.jca.workmanagers.*KEY*.short-running-threads.*KEY*.queue-size

The queue size.

thorntail.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.

thorntail.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.18. JMX

Provides support for Java Management Extensions (JMX) according to JSR-3.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jmx</artifactId>
</dependency>
```

Configuration

thorntail.jmx.audit-log-configuration.enabled

Whether audit logging is enabled.

thorntail.jmx.audit-log-configuration.log-boot

Whether operations should be logged on server boot.

thorntail.jmx.audit-log-configuration.log-read-only

Whether operations that do not modify the configuration or any runtime services should be logged.

thorntail.jmx.expression-expose-model.domain-name

The domain name to use for the 'expression' model controller JMX facade in the MBeanServer.

thorntail.jmx.jmx-remoting-connector.use-management-endpoint

If true the connector will use the management endpoint, otherwise it will use the remoting subsystem one

thorntail.jmx.non-core-mbean-sensitivity

Whether or not core MBeans, i.e. mbeans not coming from the model controller, should be considered sensitive.

thorntail.jmx.resolved-expose-model.domain-name

The domain name to use for the 'resolved' model controller JMX facade in the MBeanServer.

thorntail.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.

thorntail.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.19. JPA

Provides support for the Java Persistence API according to JSR-220.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jpa</artifactId>
</dependency>
```

Configuration

thorntail.jpa.default-datasource

The name of the default global datasource.

thorntail.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.20. JSF

Provides support for JavaServer Faces 2.3 according to JSR-372.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jsf</artifactId>
</dependency>
```

Configuration

thorntail.jsf.default-jsf-impl-slot

Default JSF implementation slot

thorntail.jsf.disallow-doctype-decl

Specifies whether or not DOCTYPE declarations in JSF deployments should be disallowed. This setting can be overridden at the deployment level.

D.21. JSON-B

Provides support for JSON Binding according to JSR-367.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jsonb</artifactId>
</dependency>
```

D.22. JSON-P

Provides support for JSON Processing according to JSR-353.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jsonp</artifactId>
</dependency>
```

D.23. KEYCLOAK

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>keycloak</artifactId>
</dependency>
```

Configuration

thorntail.keycloak.json.path

Path to Keycloak adapter configuration

thorntail.keycloak.multitenancy.paths

Map of the relative request paths to Keycloak adapter configuration locations

thorntail.keycloak.realms.*KEY*.allow-any-hostname

SSL Setting

thorntail.keycloak.realms.*KEY*.always-refresh-token

Refresh token on every single web request

thorntail.keycloak.realms.*KEY*.auth-server-url

Base URL of the Realm Auth Server

thorntail.keycloak.realms.*KEY*.auth-server-url-for-backend-requests

URL to use to make background calls to auth server

thorntail.keycloak.realms.*KEY*.autodetect-bearer-only

autodetect bearer-only requests

thorntail.keycloak.realms.*KEY*.client-key-password

n/a

thorntail.keycloak.realms.*KEY*.client-keystore

n/a

thorntail.keycloak.realms.*KEY*.client-keystore-password

n/a

thorntail.keycloak.realms.*KEY*.confidential-port

Specify the confidential port (SSL/TLS) used by the Realm Auth Server

thorntail.keycloak.realms.*KEY*.connection-pool-size

Connection pool size for the client used by the adapter

thorntail.keycloak.realms.*KEY*.cors-allowed-headers

CORS allowed headers

thorntail.keycloak.realms.*KEY*.cors-allowed-methods

CORS allowed methods

thorntail.keycloak.realms.*KEY*.cors-exposed-headers

CORS exposed headers

thorntail.keycloak.realms.*KEY*.cors-max-age

CORS max-age header

thorntail.keycloak.realms.*KEY*.disable-trust-manager

Adapter will not use a trust manager when making adapter HTTPS requests

thorntail.keycloak.realms.*KEY*.enable-cors

Enable Keycloak CORS support

thorntail.keycloak.realms.*KEY*.expose-token

Enable secure URL that exposes access token

thorntail.keycloak.realms.*KEY*.ignore-oauth-query-parameter

disable query parameter parsing for access_token

thorntail.keycloak.realms.*KEY*.principal-attribute

token attribute to use to set Principal name

thorntail.keycloak.realms.*KEY*.proxy-url

The URL for the HTTP proxy if one is used.

thorntail.keycloak.realms.*KEY*.realm-public-key

Public key of the realm

thorntail.keycloak.realms.*KEY*.register-node-at-startup

Cluster setting

thorntail.keycloak.realms.*KEY*.register-node-period

how often to re-register node

thorntail.keycloak.realms.*KEY*.ssl-required

Specify if SSL is required (valid values are all, external and none)

thorntail.keycloak.realms.*KEY*.token-store

cookie or session storage for auth session data

thorntail.keycloak.realms.*KEY*.truststore

Truststore used for adapter client HTTPS requests

thorntail.keycloak.realms.*KEY*.truststore-password

Password of the Truststore

thorntail.keycloak.realms.*KEY*.verify-token-audience

If true, then during bearer-only authentication, the adapter will verify if token contains this client name (resource) as an audience

thorntail.keycloak.secure-deployments.*KEY*.adapter-state-cookie-path

If set, defines the path used in cookies set by the adapter. Useful when deploying the application in the root context path.

thorntail.keycloak.secure-deployments.*KEY*.allow-any-hostname

SSL Setting

thorntail.keycloak.secure-deployments.*KEY*.always-refresh-token

Refresh token on every single web request

thorntail.keycloak.secure-deployments.*KEY*.auth-server-url

Base URL of the Realm Auth Server

thorntail.keycloak.secure-deployments.*KEY*.auth-server-url-for-backend-requests

URL to use to make background calls to auth server

thorntail.keycloak.secure-deployments.*KEY*.autodetect-bearer-only

autodetect bearer-only requests

thorntail.keycloak.secure-deployments.*KEY*.bearer-only

Bearer Token Auth only

thorntail.keycloak.secure-deployments.*KEY*.client-key-password

n/a

thorntail.keycloak.secure-deployments.*KEY*.client-keystore

n/a

thorntail.keycloak.secure-deployments.*KEY*.client-keystore-password

n/a

thorntail.keycloak.secure-deployments.*KEY*.confidential-port

Specify the confidential port (SSL/TLS) used by the Realm Auth Server

thorntail.keycloak.secure-deployments.*KEY*.connection-pool-size

Connection pool size for the client used by the adapter

thorntail.keycloak.secure-deployments.*KEY*.cors-allowed-headers

CORS allowed headers

thorntail.keycloak.secure-deployments.*KEY*.cors-allowed-methods

CORS allowed methods

thorntail.keycloak.secure-deployments.*KEY*.cors-exposed-headers

CORS exposed headers

thorntail.keycloak.secure-deployments.*KEY*.cors-max-age

CORS max-age header

thorntail.keycloak.secure-deployments.*KEY*.credentials.*KEY*.value

Credential value

thorntail.keycloak.secure-deployments.*KEY*.disable-trust-manager

Adapter will not use a trust manager when making adapter HTTPS requests

thorntail.keycloak.secure-deployments.*KEY*.enable-basic-auth

Enable Basic Authentication

thorntail.keycloak.secure-deployments.*KEY*.enable-cors

Enable Keycloak CORS support

thorntail.keycloak.secure-deployments.*KEY*.expose-token

Enable secure URL that exposes access token

thorntail.keycloak.secure-deployments.*KEY*.ignore-oauth-query-parameter

disable query parameter parsing for access_token

thorntail.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

thorntail.keycloak.secure-deployments.*KEY*.principal-attribute

token attribute to use to set Principal name

thorntail.keycloak.secure-deployments.*KEY*.proxy-url

The URL for the HTTP proxy if one is used.

thorntail.keycloak.secure-deployments.*KEY*.public-client

Public client

thorntail.keycloak.secure-deployments.*KEY*.public-key-cache-ttl

Maximum time the downloaded public keys are considered valid. When this time reach, the adapter is forced to download public keys from keycloak server

thorntail.keycloak.secure-deployments.*KEY*.realm

Keycloak realm

thorntail.keycloak.secure-deployments.*KEY*.realm-public-key

Public key of the realm

thorntail.keycloak.secure-deployments.*KEY*.redirect-rewrite-rules.*KEY*.value

redirect-rewrite-rule value

thorntail.keycloak.secure-deployments.*KEY*.register-node-at-startup

Cluster setting

thorntail.keycloak.secure-deployments.*KEY*.register-node-period

how often to re-register node

thorntail.keycloak.secure-deployments.*KEY*.resource

Application name

thorntail.keycloak.secure-deployments.*KEY*.ssl-required

Specify if SSL is required (valid values are all, external and none)

thorntail.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

thorntail.keycloak.secure-deployments.*KEY*.token-store

cookie or session storage for auth session data

thorntail.keycloak.secure-deployments.*KEY*.truststore

Truststore used for adapter client HTTPS requests

thorntail.keycloak.secure-deployments.*KEY*.truststore-password

Password of the Truststore

thorntail.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

thorntail.keycloak.secure-deployments.*KEY*.use-resource-role-mappings

Use resource level permissions from token

thorntail.keycloak.secure-deployments.*KEY*.verify-token-audience

If true, then during bearer-only authentication, the adapter will verify if token contains this client name (resource) as an audience

thorntail.keycloak.secure-servers.*KEY*.adapter-state-cookie-path

If set, defines the path used in cookies set by the adapter. Useful when deploying the application in the root context path.

thorntail.keycloak.secure-servers.*KEY*.allow-any-hostname

SSL Setting

thorntail.keycloak.secure-servers.*KEY*.always-refresh-token

Refresh token on every single web request

thorntail.keycloak.secure-servers.*KEY*.auth-server-url

Base URL of the Realm Auth Server

thorntail.keycloak.secure-servers.*KEY*.auth-server-url-for-backend-requests

URL to use to make background calls to auth server

thorntail.keycloak.secure-servers.*KEY*.autodetect-bearer-only

autodetect bearer-only requests

thorntail.keycloak.secure-servers.*KEY*.bearer-only

Bearer Token Auth only

thorntail.keycloak.secure-servers.*KEY*.client-key-password

n/a

thorntail.keycloak.secure-servers.*KEY*.client-keystore

n/a

thorntail.keycloak.secure-servers.*KEY*.client-keystore-password

n/a

thorntail.keycloak.secure-servers.*KEY*.confidential-port

Specify the confidential port (SSL/TLS) used by the Realm Auth Server

thorntail.keycloak.secure-servers.*KEY*.connection-pool-size

Connection pool size for the client used by the adapter

thorntail.keycloak.secure-servers.*KEY*.cors-allowed-headers

CORS allowed headers

thorntail.keycloak.secure-servers.*KEY*.cors-allowed-methods

CORS allowed methods

thorntail.keycloak.secure-servers.*KEY*.cors-exposed-headers

CORS exposed headers

thorntail.keycloak.secure-servers.*KEY*.cors-max-age

CORS max-age header

thorntail.keycloak.secure-servers.*KEY*.credentials.*KEY*.value

Credential value

thorntail.keycloak.secure-servers.*KEY*.disable-trust-manager

Adapter will not use a trust manager when making adapter HTTPS requests

thorntail.keycloak.secure-servers.*KEY*.enable-basic-auth

Enable Basic Authentication

thorntail.keycloak.secure-servers.*KEY*.enable-cors

Enable Keycloak CORS support

thorntail.keycloak.secure-servers.*KEY*.expose-token

Enable secure URL that exposes access token

thorntail.keycloak.secure-servers.*KEY*.ignore-oauth-query-parameter

disable query parameter parsing for access_token

thorntail.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

thorntail.keycloak.secure-servers.*KEY*.principal-attribute

token attribute to use to set Principal name

thorntail.keycloak.secure-servers.*KEY*.proxy-url

The URL for the HTTP proxy if one is used.

thorntail.keycloak.secure-servers.*KEY*.public-client

Public client

thorntail.keycloak.secure-servers.*KEY*.public-key-cache-ttl

Maximum time the downloaded public keys are considered valid. When this time reach, the adapter is forced to download public keys from keycloak server

thorntail.keycloak.secure-servers.*KEY*.realm

Keycloak realm

thorntail.keycloak.secure-servers.*KEY*.realm-public-key

Public key of the realm

thorntail.keycloak.secure-servers.*KEY*.redirect-rewrite-rules.*KEY*.value

redirect-rewrite-rule value

thorntail.keycloak.secure-servers.*KEY*.register-node-at-startup

Cluster setting

thorntail.keycloak.secure-servers.*KEY*.register-node-period

how often to re-register node

thorntail.keycloak.secure-servers.*KEY*.resource

Application name

thorntail.keycloak.secure-servers.*KEY*.ssl-required

Specify if SSL is required (valid values are all, external and none)

thorntail.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

thorntail.keycloak.secure-servers.*KEY*.token-store

cookie or session storage for auth session data

thorntail.keycloak.secure-servers.*KEY*.truststore

Truststore used for adapter client HTTPS requests

thorntail.keycloak.secure-servers.*KEY*.truststore-password

Password of the Truststore

thorntail.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

thorntail.keycloak.secure-servers.*KEY*.use-resource-role-mappings

Use resource level permissions from token

thorntail.keycloak.secure-servers.*KEY*.verify-token-audience

If true, then during bearer-only authentication, the adapter will verify if token contains this client name (resource) as an audience

D.24. 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 **thorntail.logging.loggers.[com.mycorp.logger].level**.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>logging</artifactId>
</dependency>
```

Configuration

thorntail.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.

thorntail.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.

thorntail.logging.async-handlers.*KEY*.filter

Defines a simple filter type.

thorntail.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.*"))

thorntail.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.

thorntail.logging.async-handlers.*KEY*.name

The name of the handler.

thorntail.logging.async-handlers.*KEY*.overflow-action

Specify what action to take when the overflowing. The valid options are 'block' and 'discard'

thorntail.logging.async-handlers.*KEY*.queue-length

The queue length to use before flushing writing

thorntail.logging.async-handlers.*KEY*.subhandlers

The Handlers associated with this async handler.

thorntail.logging.console-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.console-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.console-handlers.*KEY*.filter

Defines a simple filter type.

thorntail.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.*"))

thorntail.logging.console-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.console-handlers.*KEY*.name

The name of the handler.

thorntail.logging.console-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.logging.console-handlers.*KEY*.target

Defines the target of the console handler. The value can be System.out, System.err or console.

thorntail.logging.custom-formatters.*KEY*.attribute-class

The logging formatter class to be used.

thorntail.logging.custom-formatters.*KEY*.module

The module that the logging formatter depends on.

thorntail.logging.custom-formatters.*KEY*.properties

Defines the properties used for the logging formatter. All properties must be accessible via a setter method.

thorntail.logging.custom-handlers.*KEY*.attribute-class

The logging handler class to be used.

thorntail.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.

thorntail.logging.custom-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.custom-handlers.*KEY*.filter

Defines a simple filter type.

thorntail.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.*"))

thorntail.logging.custom-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.custom-handlers.*KEY*.module

The module that the logging handler depends on.

thorntail.logging.custom-handlers.*KEY*.name

The name of the handler.

thorntail.logging.custom-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.logging.custom-handlers.*KEY*.properties

Defines the properties used for the logging handler. All properties must be accessible via a setter method.

thorntail.logging.file-handlers.*KEY*.append

Specify whether to append to the target file.

thorntail.logging.file-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.file-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

thorntail.logging.file-handlers.*KEY*.filter

Defines a simple filter type.

thorntail.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.*"))

thorntail.logging.file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.file-handlers.*KEY*.name

The name of the handler.

thorntail.logging.file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.logging.filters.*KEY*.attribute-class

The logging filter class to be used.

thorntail.logging.filters.*KEY*.constructor-properties

Defines the constructor properties used for the logging filter.

thorntail.logging.filters.*KEY*.module

The module that the logging filter depends on.

thorntail.logging.filters.*KEY*.properties

Defines the properties used for the logging filter. All properties must be accessible via a setter method.

thorntail.logging.json-formatters.*KEY*.date-format

The date/time format pattern. The pattern must be a valid java.time.format.DateTimeFormatter.ofPattern() pattern. The default pattern is an ISO-8601 extended offset date-time format.

thorntail.logging.json-formatters.*KEY*.exception-output-type

Indicates how the cause of the logged message, if one is available, will be added to the JSON output.

thorntail.logging.json-formatters.*KEY*.key-overrides

Allows the names of the keys for the JSON properties to be overridden.

thorntail.logging.json-formatters.*KEY*.meta-data

Sets the meta data to use in the JSON format. Properties will be added to each log message.

thorntail.logging.json-formatters.*KEY*.pretty-print

Indicates whether or not pretty printing should be used when formatting.

thorntail.logging.json-formatters.*KEY*.print-details

Sets whether or not details should be printed. Printing the details can be expensive as the values are retrieved from the caller. The details include the source class name, source file name, source method name, source module name, source module version and source line number.

thorntail.logging.json-formatters.*KEY*.record-delimiter

The value to be used to indicate the end of a record. If set to null no delimiter will be used at the end of the record. The default value is a line feed.

thorntail.logging.json-formatters.*KEY*.zone-id

The zone ID for formatting the date and time. The system default is used if left undefined.

thorntail.logging.log-files.*KEY*.file-size

The size of the log file in bytes.

thorntail.logging.log-files.*KEY*.last-modified-time

The date, in milliseconds, the file was last modified.

thorntail.logging.log-files.*KEY*.last-modified-timestamp

The date, in ISO 8601 format, the file was last modified.

thorntail.logging.log-files.*KEY*.stream

Provides the server log as a response attachment. The response result value is the unique id of the attachment.

thorntail.logging.loggers.*KEY*.category

Specifies the category for the logger.

thorntail.logging.loggers.*KEY*.filter

Defines a simple filter type.

thorntail.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.*"))

thorntail.logging.loggers.*KEY*.handlers

The handlers associated with the logger.

thorntail.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.

thorntail.logging.loggers.*KEY*.use-parent-handlers

Specifies whether or not this logger should send its output to its parent Logger.

thorntail.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.

thorntail.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.*"))

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.async-handlers.*KEY*.name

The name of the handler.

thorntail.logging.logging-profiles.*KEY*.async-handlers.*KEY*.overflow-action

Specify what action to take when the overflowing. The valid options are 'block' and 'discard'

thorntail.logging.logging-profiles.*KEY*.async-handlers.*KEY*.queue-length

The queue length to use before flushing writing

thorntail.logging.logging-profiles.*KEY*.async-handlers.*KEY*.subhandlers

The Handlers associated with this async handler.

thorntail.logging.logging-profiles.*KEY*.console-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.console-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.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.*"))

thorntail.logging.logging-profiles.*KEY*.console-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.console-handlers.*KEY*.name

The name of the handler.

thorntail.logging.logging-profiles.*KEY*.console-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.custom-formatters.*KEY*.attribute-class

The logging formatter class to be used.

thorntail.logging.logging-profiles.*KEY*.custom-formatters.*KEY*.module

The module that the logging formatter depends on.

thorntail.logging.logging-profiles.*KEY*.custom-formatters.*KEY*.properties

Defines the properties used for the logging formatter. All properties must be accessible via a setter method.

thorntail.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.attribute-class

The logging handler class to be used.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.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.*"))

thorntail.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.module

The module that the logging handler depends on.

thorntail.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.name

The name of the handler.

thorntail.logging.logging-profiles.*KEY*.custom-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.file-handlers.*KEY*.append

Specify whether to append to the target file.

thorntail.logging.logging-profiles.*KEY*.file-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.file-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.logging-profiles.*KEY*.file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

thorntail.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.*"))

thorntail.logging.logging-profiles.*KEY*.file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.file-handlers.*KEY*.name

The name of the handler.

thorntail.logging.logging-profiles.*KEY*.file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.logging.logging-profiles.*KEY*.filters.*KEY*.attribute-class

The logging filter class to be used.

thorntail.logging.logging-profiles.*KEY*.filters.*KEY*.constructor-properties

Defines the constructor properties used for the logging filter.

thorntail.logging.logging-profiles.*KEY*.filters.*KEY*.module

The module that the logging filter depends on.

thorntail.logging.logging-profiles.*KEY*.filters.*KEY*.properties

Defines the properties used for the logging filter. All properties must be accessible via a setter method.

thorntail.logging.logging-profiles.*KEY*.json-formatters.*KEY*.date-format

The date/time format pattern. The pattern must be a valid java.time.format.DateTimeFormatter.ofPattern() pattern. The default pattern is an ISO-8601 extended offset date-time format.

thorntail.logging.logging-profiles.*KEY*.json-formatters.*KEY*.exception-output-type

Indicates how the cause of the logged message, if one is available, will be added to the JSON output.

thorntail.logging.logging-profiles.*KEY*.json-formatters.*KEY*.key-overrides

Allows the names of the keys for the JSON properties to be overridden.

thorntail.logging.logging-profiles.*KEY*.json-formatters.*KEY*.meta-data

Sets the meta data to use in the JSON format. Properties will be added to each log message.

thorntail.logging.logging-profiles.*KEY*.json-formatters.*KEY*.pretty-print

Indicates whether or not pretty printing should be used when formatting.

thorntail.logging.logging-profiles.*KEY*.json-formatters.*KEY*.print-details

Sets whether or not details should be printed. Printing the details can be expensive as the values are retrieved from the caller. The details include the source class name, source file name, source method name, source module name, source module version and source line number.

thorntail.logging.logging-profiles.*KEY*.json-formatters.*KEY*.record-delimiter

The value to be used to indicate the end of a record. If set to null no delimiter will be used at the end of the record. The default value is a line feed.

thorntail.logging.logging-profiles.*KEY*.json-formatters.*KEY*.zone-id

The zone ID for formatting the date and time. The system default is used if left undefined.

thorntail.logging.logging-profiles.*KEY*.log-files.*KEY*.file-size

The size of the log file in bytes.

thorntail.logging.logging-profiles.*KEY*.log-files.*KEY*.last-modified-time

The date, in milliseconds, the file was last modified.

thorntail.logging.logging-profiles.*KEY*.log-files.*KEY*.last-modified-timestamp

The date, in ISO 8601 format, the file was last modified.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.loggers.*KEY*.category

Specifies the category for the logger.

thorntail.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.*"))

thorntail.logging.logging-profiles.*KEY*.loggers.*KEY*.handlers

The handlers associated with the logger.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.loggers.*KEY*.use-parent-handlers

Specifies whether or not this logger should send its output to its parent Logger.

thorntail.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

thorntail.logging.logging-profiles.*KEY*.pattern-formatters.*KEY*.pattern

Defines a pattern for the formatter.

thorntail.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

thorntail.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

thorntail.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.*"))

thorntail.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.name

The name of the handler.

thorntail.logging.logging-profiles.*KEY*.periodic-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

thorntail.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.*"))

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.name

The name of the handler.

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.periodic-size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

thorntail.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.

thorntail.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.*"))

thorntail.logging.logging-profiles.*KEY*.root-logger.handlers

The handlers associated with the root logger.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

thorntail.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.*"))

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.name

The name of the handler.

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.socket-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.logging.logging-profiles.*KEY*.socket-handlers.*KEY*.block-on-reconnect

If set to true the write methods will block when attempting to reconnect. This is only advisable to be set to true if using an asynchronous handler.

thorntail.logging.logging-profiles.*KEY*.socket-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.

thorntail.logging.logging-profiles.*KEY*.socket-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.logging-profiles.*KEY*.socket-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.*"))

thorntail.logging.logging-profiles.*KEY*.socket-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.

thorntail.logging.logging-profiles.*KEY*.socket-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.logging.logging-profiles.*KEY*.socket-handlers.*KEY*.outbound-socket-binding-ref

Outbound socket reference for the socket connection.

thorntail.logging.logging-profiles.*KEY*.socket-handlers.*KEY*.protocol

The protocol the socket should communicate over.

thorntail.logging.logging-profiles.*KEY*.socket-handlers.*KEY*.ssl-context

The reference to the defined SSL context. This is only used if the protocol is set to SSL_TCP.

thorntail.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".

thorntail.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.

thorntail.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).

thorntail.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.

thorntail.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.

thorntail.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.port

The port the syslog server is listening on.

thorntail.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.server-address

The address of the syslog server.

thorntail.logging.logging-profiles.*KEY*.syslog-handlers.*KEY*.syslog-format

Formats the log message according to the RFC specification.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.date-format

The date/time format pattern. The pattern must be a valid java.time.format.DateTimeFormatter.ofPattern() pattern. The default pattern is an ISO-8601 extended offset date-time format.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.exception-output-type

Indicates how the cause of the logged message, if one is available, will be added to the XML output.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.key-overrides

Allows the names of the keys for the XML properties to be overridden.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.meta-data

Sets the meta data to use in the XML format. Properties will be added to each log message.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.namespace-uri

Sets the namespace URI used for each record if print-namespace attribute is true. Note that if no namespace-uri is defined and there are overridden keys no namespace will be written regardless if the print-namespace attribute is set to true.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.pretty-print

Indicates whether or not pretty printing should be used when formatting.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.print-details

Sets whether or not details should be printed. Printing the details can be expensive as the values are retrieved from the caller. The details include the source class name, source file name, source method name, source module name, source module version and source line number.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.print-namespace

Turns on or off the printing of the namespace for each <record/>. This is set to false by default.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.record-delimiter

The value to be used to indicate the end of a record. If set to null no delimiter will be used at the end of the record. The default value is a line feed.

thorntail.logging.logging-profiles.*KEY*.xml-formatters.*KEY*.zone-id

The zone ID for formatting the date and time. The system default is used if left undefined.

thorntail.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

thorntail.logging.pattern-formatters.*KEY*.pattern

Defines a pattern for the formatter.

thorntail.logging.periodic-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

thorntail.logging.periodic-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.periodic-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.periodic-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

thorntail.logging.periodic-rotating-file-handlers.*KEY*.filter

Defines a simple filter type.

thorntail.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.*"))

thorntail.logging.periodic-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.periodic-rotating-file-handlers.*KEY*.name

The name of the handler.

thorntail.logging.periodic-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.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.

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

thorntail.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.*"))

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.name

The name of the handler.

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.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.

thorntail.logging.periodic-size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

thorntail.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.

thorntail.logging.root-logger.filter

Defines a simple filter type.

thorntail.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.*"))

thorntail.logging.root-logger.handlers

The handlers associated with the root logger.

thorntail.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.

thorntail.logging.size-rotating-file-handlers.*KEY*.append

Specify whether to append to the target file.

thorntail.logging.size-rotating-file-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.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.

thorntail.logging.size-rotating-file-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.size-rotating-file-handlers.*KEY*.file

The file description consisting of the path and optional relative to path.

thorntail.logging.size-rotating-file-handlers.*KEY*.filter

Defines a simple filter type.

thorntail.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.*"))

thorntail.logging.size-rotating-file-handlers.*KEY*.formatter

Defines a pattern for the formatter.

thorntail.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.

thorntail.logging.size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

thorntail.logging.size-rotating-file-handlers.*KEY*.name

The name of the handler.

thorntail.logging.size-rotating-file-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.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.

thorntail.logging.size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

thorntail.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.

thorntail.logging.socket-handlers.*KEY*.autoflush

Automatically flush after each write.

thorntail.logging.socket-handlers.*KEY*.block-on-reconnect

If set to true the write methods will block when attempting to reconnect. This is only advisable to be set to true if using an asynchronous handler.

thorntail.logging.socket-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.

thorntail.logging.socket-handlers.*KEY*.encoding

The character encoding used by this Handler.

thorntail.logging.socket-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.*"))

thorntail.logging.socket-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.

thorntail.logging.socket-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.logging.socket-handlers.*KEY*.outbound-socket-binding-ref

Outbound socket reference for the socket connection.

thorntail.logging.socket-handlers.*KEY*.protocol

The protocol the socket should communicate over.

thorntail.logging.socket-handlers.*KEY*.ssl-context

The reference to the defined SSL context. This is only used if the protocol is set to SSL_TCP.

thorntail.logging.syslog-handlers.*KEY*.app-name

The app name used when formatting the message in RFC5424 format. By default the app name is "java".

thorntail.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.

thorntail.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).

thorntail.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.

thorntail.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.

thorntail.logging.syslog-handlers.*KEY*.named-formatter

The name of the defined formatter to be used on the handler.

thorntail.logging.syslog-handlers.*KEY*.port

The port the syslog server is listening on.

thorntail.logging.syslog-handlers.*KEY*.server-address

The address of the syslog server.

thorntail.logging.syslog-handlers.*KEY*.syslog-format

Formats the log message according to the RFC specification.

thorntail.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.

thorntail.logging.xml-formatters.*KEY*.date-format

The date/time format pattern. The pattern must be a valid java.time.format.DateTimeFormatter.ofPattern() pattern. The default pattern is an ISO-8601 extended offset date-time format.

thorntail.logging.xml-formatters.*KEY*.exception-output-type

Indicates how the cause of the logged message, if one is available, will be added to the XML output.

thorntail.logging.xml-formatters.*KEY*.key-overrides

Allows the names of the keys for the XML properties to be overridden.

thorntail.logging.xml-formatters.*KEY*.meta-data

Sets the meta data to use in the XML format. Properties will be added to each log message.

thorntail.logging.xml-formatters.*KEY*.namespace-uri

Sets the namespace URI used for each record if print-namespace attribute is true. Note that if no namespace-uri is defined and there are overridden keys no namespace will be written regardless if the print-namespace attribute is set to true.

thorntail.logging.xml-formatters.*KEY*.pretty-print

Indicates whether or not pretty printing should be used when formatting.

thorntail.logging.xml-formatters.*KEY*.print-details

Sets whether or not details should be printed. Printing the details can be expensive as the values are retrieved from the caller. The details include the source class name, source file name, source method name, source module name, source module version and source line number.

thorntail.logging.xml-formatters.*KEY*.print-namespace

Turns on or off the printing of the namespace for each <record/>. This is set to false by default.

thorntail.logging.xml-formatters.*KEY*.record-delimiter

The value to be used to indicate the end of a record. If set to null no delimiter will be used at the end of the record. The default value is a line feed.

thorntail.logging.xml-formatters.*KEY*.zone-id

The zone ID for formatting the date and time. The system default is used if left undefined.

D.25. MANAGEMENT

Provides the JBoss EAP management API.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>management</artifactId>
</dependency>
```

Configuration

thorntail.management.audit-access.audit-log-logger.enabled

Whether audit logging is enabled.

thorntail.management.audit-access.audit-log-logger.log-boot

Whether operations should be logged on server boot.

thorntail.management.audit-access.audit-log-logger.log-read-only

Whether operations that do not modify the configuration or any runtime services should be logged.

thorntail.management.audit-access.file-handlers.*KEY*.disabled-due-to-failure

Whether this handler has been disabled due to logging failures.

thorntail.management.audit-access.file-handlers.*KEY*.failure-count

The number of logging failures since the handler was initialized.

thorntail.management.audit-access.file-handlers.*KEY*.formatter

The formatter used to format the log messages.

thorntail.management.audit-access.file-handlers.*KEY*.max-failure-count

The maximum number of logging failures before disabling this handler.

thorntail.management.audit-access.file-handlers.*KEY*.path

The path of the audit log file.

thorntail.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.

thorntail.management.audit-access.file-handlers.*KEY*.rotate-at-startup

Whether the old log file should be rotated at server startup.

thorntail.management.audit-access.in-memory-handlers.*KEY*.max-history

The maximum number of operation stored in history for this handler.

thorntail.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.

thorntail.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".

thorntail.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".

thorntail.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".

thorntail.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".

thorntail.management.audit-access.json-formatters.*KEY*.include-date

Whether or not to include the date in the formatted log record.

thorntail.management.audit-access.periodic-rotating-file-handlers.*KEY*.disabled-due-to-failure

Whether this handler has been disabled due to logging failures.

thorntail.management.audit-access.periodic-rotating-file-handlers.*KEY*.failure-count

The number of logging failures since the handler was initialized.

thorntail.management.audit-access.periodic-rotating-file-handlers.*KEY*.formatter

The formatter used to format the log messages.

thorntail.management.audit-access.periodic-rotating-file-handlers.*KEY*.max-failure-count

The maximum number of logging failures before disabling this handler.

thorntail.management.audit-access.periodic-rotating-file-handlers.*KEY*.path

The path of the audit log file.

thorntail.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.

thorntail.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.

thorntail.management.audit-access.size-rotating-file-handlers.*KEY*.disabled-due-to-failure

Whether this handler has been disabled due to logging failures.

thorntail.management.audit-access.size-rotating-file-handlers.*KEY*.failure-count

The number of logging failures since the handler was initialized.

thorntail.management.audit-access.size-rotating-file-handlers.*KEY*.formatter

The formatter used to format the log messages.

thorntail.management.audit-access.size-rotating-file-handlers.*KEY*.max-backup-index

The maximum number of backups to keep.

thorntail.management.audit-access.size-rotating-file-handlers.*KEY*.max-failure-count

The maximum number of logging failures before disabling this handler.

thorntail.management.audit-access.size-rotating-file-handlers.*KEY*.path

The path of the audit log file.

thorntail.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.

thorntail.management.audit-access.size-rotating-file-handlers.*KEY*.rotate-size

The size at which to rotate the log file.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.disabled-due-to-failure

Whether this handler has been disabled due to logging failures.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.failure-count

The number of logging failures since the handler was initialized.

thorntail.management.audit-access.syslog-handlers.*KEY*.formatter

The formatter used to format the log messages.

thorntail.management.audit-access.syslog-handlers.*KEY*.max-failure-count

The maximum number of logging failures before disabling this handler.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.syslog-format

Whether to set the syslog format to the one specified in RFC-5424 or RFC-3164.

thorntail.management.audit-access.syslog-handlers.*KEY*.tcp-protocol.host

The host of the syslog server for the tcp requests.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.tcp-protocol.port

The port of the syslog server for the tcp requests.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.key-password

The password for the keystore key.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.keystore-password

The password for the keystore.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.tls-protocol.client-certificate-store-authentication.keystore-path

The path of the keystore.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.tls-protocol.host

The host of the syslog server for the tls over tcp requests.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.tls-protocol.port

The port of the syslog server for the tls over tcp requests.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.tls-protocol.truststore-authentication.keystore-password

The password for the truststore.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.tls-protocol.truststore-authentication.keystore-path

The path of the truststore.

thorntail.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.

thorntail.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.

thorntail.management.audit-access.syslog-handlers.*KEY*.udp-protocol.host

The host of the syslog server for the udp requests.

thorntail.management.audit-access.syslog-handlers.*KEY*.udp-protocol.port

The port of the syslog server for the udp requests.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.management.authorization-access.application-classification-constraint.types.*KEY*.classifications.*KEY*.default-application

Whether targets having this application type constraint are considered application resources.

thorntail.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.

thorntail.management.authorization-access.provider

The provider to use for management access control decisions.

thorntail.management.authorization-access.role-mappings.*KEY*.excludes.*KEY*.name

The name of the user or group being mapped.

thorntail.management.authorization-access.role-mappings.*KEY*.excludes.*KEY*.realm

An optional attribute to map based on the realm used for authentication.

thorntail.management.authorization-access.role-mappings.*KEY*.excludes.*KEY*.type

The type of the Principal being mapped, either 'group' or 'user'.

thorntail.management.authorization-access.role-mappings.*KEY*.include-all

Configure if all authenticated users should be automatically assigned this role.

thorntail.management.authorization-access.role-mappings.*KEY*.includes.*KEY*.name

The name of the user or group being mapped.

thorntail.management.authorization-access.role-mappings.*KEY*.includes.*KEY*.realm

An optional attribute to map based on the realm used for authentication.

thorntail.management.authorization-access.role-mappings.*KEY*.includes.*KEY*.type

The type of the Principal being mapped, either 'group' or 'user'.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.management.authorization-access.sensitivity-classification-constraint.types.*KEY*.classifications.*KEY*.default-application

Whether targets having this application type constraint are considered application resources.

thorntail.management.authorization-access.standard-role-names

The official names of the standard roles supported by the current management access control provider.

thorntail.management.authorization-access.use-identity-roles

Should the raw roles obtained from the underlying security identity be used directly?

thorntail.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.

thorntail.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.

thorntail.management.authorization-access.vault-expression-constraint.default-requires-read

Whether reading attributes containing vault expressions should be considered sensitive.

thorntail.management.authorization-access.vault-expression-constraint.default-requires-write

Whether writing attributes containing vault expressions should be considered sensitive.

thorntail.management.bind.interface

Interface to bind for the management ports

thorntail.management.configuration-changes-service.max-history

The maximum number of configuration changes stored in history.

thorntail.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.

thorntail.management.http-interface-management-interface.console-enabled

Flag that indicates admin console is enabled

thorntail.management.http-interface-management-interface.constant-headers

The set of constant HTTP headers to apply to response messages.

thorntail.management.http-interface-management-interface.http-authentication-factory

The authentication policy to use to secure the interface for normal HTTP requests.

thorntail.management.http-interface-management-interface.http-upgrade

HTTP Upgrade specific configuration

thorntail.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

thorntail.management.http-interface-management-interface.sasl-protocol

The name of the protocol to be passed to the SASL mechanisms used for authentication.

thorntail.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.

thorntail.management.http-interface-management-interface.security-realm

The legacy security realm to use for the HTTP management interface.

thorntail.management.http-interface-management-interface.server-name

The name of the server used in the initial Remoting exchange and within the SASL mechanisms.

thorntail.management.http-interface-management-interface.socket-binding

The name of the socket binding configuration to use for the HTTP management interface’s socket.

thorntail.management.http-interface-management-interface.ssl-context

Reference to the SSLContext to use for this management interface.

thorntail.management.http.disable

Flag to disable HTTP access to management interface

thorntail.management.http.port

Port for HTTP access to management interface

thorntail.management.https.port

Port for HTTPS access to management interface

thorntail.management.identity-access.security-domain

Reference to the security domain to use to obtain the current identity performing a management request.

thorntail.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

thorntail.management.ldap-connections.*KEY*.handles-referrals-for

List of URLs that this connection handles referrals for.

thorntail.management.ldap-connections.*KEY*.initial-context-factory

The initial context factory to establish the LdapContext.

thorntail.management.ldap-connections.*KEY*.properties.*KEY*.value

The optional value of the property.

thorntail.management.ldap-connections.*KEY*.referrals

The referral handling mode for this connection.

thorntail.management.ldap-connections.*KEY*.search-credential

The credential to use when connecting to perform a search.

thorntail.management.ldap-connections.*KEY*.search-credential-reference

The reference to the search credential stored in CredentialStore under defined alias or clear text password.

thorntail.management.ldap-connections.*KEY*.search-dn

The distinguished name to use when connecting to the LDAP server to perform searches.

thorntail.management.ldap-connections.*KEY*.security-realm

The security realm to reference to obtain a configured SSLContext to use when establishing the connection.

thorntail.management.ldap-connections.*KEY*.url

The URL to use to connect to the LDAP server.

thorntail.management.management-operations-service.active-operations.*KEY*.access-mechanism

The mechanism used to submit a request to the server.

thorntail.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.

thorntail.management.management-operations-service.active-operations.*KEY*.caller-thread

The name of the thread that is executing the operation.

thorntail.management.management-operations-service.active-operations.*KEY*.cancelled

Whether the operation has been cancelled.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.management.management-operations-service.active-operations.*KEY*.execution-status

The current activity of the operation.

thorntail.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.

thorntail.management.management-operations-service.active-operations.*KEY*.running-time

Amount of time the operation has been executing.

thorntail.management.native-interface-management-interface.sasl-authentication-factory

The SASL authentication policy to use to secure this interface.

thorntail.management.native-interface-management-interface.sasl-protocol

The name of the protocol to be passed to the SASL mechanisms used for authentication.

thorntail.management.native-interface-management-interface.security-realm

The legacy security realm to use for the native management interface.

thorntail.management.native-interface-management-interface.server-name

The name of the server used in the initial Remoting exchange and within the SASL mechanisms.

thorntail.management.native-interface-management-interface.socket-binding

The name of the socket binding configuration to use for the native management interface's socket.

thorntail.management.native-interface-management-interface.ssl-context

Reference to the SSLContext to use for this management interface.

thorntail.management.security-realms.*KEY*.jaas-authentication.assign-groups

Map the roles loaded by JAAS to groups.

thorntail.management.security-realms.*KEY*.jaas-authentication.name

The name of the JAAS configuration to use.

thorntail.management.security-realms.*KEY*.kerberos-authentication.remove-realm

After authentication should the realm name be stripped from the users name.

thorntail.management.security-realms.*KEY*.kerberos-server-identity.keytabs.*KEY*.debug

Should additional debug logging be enabled during TGT acquisition?

thorntail.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.

thorntail.management.security-realms.*KEY*.kerberos-server-identity.keytabs.*KEY*.path

The path to the keytab.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authentication.allow-empty-passwords

Should empty passwords be accepted from the user being authenticated.

thorntail.management.security-realms.*KEY*.ldap-authentication.base-dn

The base distinguished name to commence the search for the user.

thorntail.management.security-realms.*KEY*.ldap-authentication.by-access-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authentication.by-access-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authentication.by-search-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authentication.by-search-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authentication.connection

The name of the connection to use to connect to LDAP.

thorntail.management.security-realms.*KEY*.ldap-authentication.recursive

Whether the search should be recursive.

thorntail.management.security-realms.*KEY*.ldap-authentication.user-dn

The name of the attribute which is the user’s distinguished name.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.base-dn

The starting point of the search for the user.

thorntail.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-access-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-access-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-search-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.by-search-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.filter

The filter to use for the LDAP search.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.advanced-filter-username-to-dn.recursive

Should levels below the starting point be recursively searched?

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.connection

The name of the connection to use to connect to LDAP.

thorntail.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.base-dn

The starting point of the search for the group.

thorntail.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-access-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-access-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-search-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.by-search-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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?

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.principal-attribute

The attribute on the group entry that references the principal.

thorntail.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.recursive

Should levels below the starting point be recursively searched?

thorntail.management.security-realms.*KEY*.ldap-authorization.group-to-principal-group-search.search-by

Should searches be performed using simple names or distinguished names?

thorntail.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-access-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-access-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-search-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.principal-to-group-group-search.by-search-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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?

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.attribute

The attribute on the user entry that is their username.

thorntail.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.base-dn

The starting point of the search for the user.

thorntail.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-access-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-access-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-search-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.by-search-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.username-filter-username-to-dn.recursive

Should levels below the starting point be recursively searched?

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-access-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-access-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-search-time-cache.cache-failures

Should failures be cached?

thorntail.management.security-realms.*KEY*.ldap-authorization.username-is-dn-username-to-dn.by-search-time-cache.cache-size

The current size of the cache.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.local-authentication.default-user

The name of the default user to assume if no user specified by the remote client.

thorntail.management.security-realms.*KEY*.local-authentication.skip-group-loading

Disable the loading of the users group membership information after local authentication has been used.

thorntail.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.

thorntail.management.security-realms.*KEY*.plug-in-authentication.mechanism

Allow the mechanism this plug-in is compatible with to be overridden from DIGEST.

thorntail.management.security-realms.*KEY*.plug-in-authentication.name

The short name of the plug-in (as registered) to use.

thorntail.management.security-realms.*KEY*.plug-in-authentication.properties.*KEY*.value

The optional value of the property.

thorntail.management.security-realms.*KEY*.plug-in-authorization.name

The short name of the plug-in (as registered) to use.

thorntail.management.security-realms.*KEY*.plug-in-authorization.properties.*KEY*.value

The optional value of the property.

thorntail.management.security-realms.*KEY*.properties-authentication.path

The path of the properties file containing the users.

thorntail.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'.

thorntail.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.

thorntail.management.security-realms.*KEY*.properties-authorization.path

The path of the properties file containing the users roles.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.secret-server-identity.value

The secret / password - Base64 Encoded.

thorntail.management.security-realms.*KEY*.ssl-server-identity.alias

The alias of the entry to use from the keystore.

thorntail.management.security-realms.*KEY*.ssl-server-identity.enabled-cipher-suites

The cipher suites that can be enabled on the underlying SSLEngine.

thorntail.management.security-realms.*KEY*.ssl-server-identity.enabled-protocols

The protocols that can be enabled on the underlying SSLEngine.

thorntail.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.

thorntail.management.security-realms.*KEY*.ssl-server-identity.key-password

The password to obtain the key from the keystore.

thorntail.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.

thorntail.management.security-realms.*KEY*.ssl-server-identity.keystore-password

The password to open the keystore.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.ssl-server-identity.keystore-provider

The provider for loading the keystore, defaults to JKS.

thorntail.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.

thorntail.management.security-realms.*KEY*.ssl-server-identity.protocol

The protocol to use when creating the SSLContext.

thorntail.management.security-realms.*KEY*.truststore-authentication.keystore-password

The password to open the keystore.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.truststore-authentication.keystore-provider

The provider for loading the keystore, defaults to JKS.

thorntail.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.

thorntail.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.

thorntail.management.security-realms.*KEY*.users-authentication.users.*KEY*.password

The user’s password.

D.26. MICROPROFILE

You can use this fraction to add a dependency on all fractions that implement the Eclipse MicroProfile specifications.

D.26.1. Note about YAML configuration

Some Eclipse MicroProfile specifications define configuration properties that use / as a delimiter, because the **.** character is used in fully qualified class names. When writing the YAML configuration, it is required to split around **.** and *not* around /.

Example D.1. YAML configuration for MicroProfile Rest Client

For example, MicroProfile Rest Client specifies that you can configure URL of an external service with a configuration property named **com.example.demo.client.Service/mp-rest/url**. This translates to the following YAML:

```
com:
  example:
    demo:
      client:
        Service/mp-rest/url: http://localhost:8080/...
```

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile</artifactId>
</dependency>
```

D.26.2. MicroProfile Config

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-config</artifactId>
</dependency>
```

Configuration

thorntail.microprofile.config.config-source-providers.*KEY*.attribute-class

Class of the ConfigSourceProvider to load

thorntail.microprofile.config.config-sources.*KEY*.attribute-class

Class of the config source to load

thorntail.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)

thorntail.microprofile.config.config-sources.*KEY*.ordinal

Ordinal value for the config source

thorntail.microprofile.config.config-sources.*KEY*.properties

Properties configured for this config source

D.26.3. MicroProfile Fault Tolerance

This fraction implements the Eclipse MicroProfile Fault Tolerance API. It is based on SmallRye Fault Tolerance.

D.26.3.1. Configuration

SmallRye Fault Tolerance supports all the configuration options of MicroProfile Fault Tolerance and, additionally, the following options:

- **io.smallrye.faulttolerance.globalThreadPoolSize**: size of the thread pool used for running **@Asynchronous** actions. This does not include bulkhead thread pools. Defaults to **100**.

- **io.smallrye.faulttolerance.timeoutExecutorThreads**: size of the thread pool used for scheduling timeouts. Defaults to **5**.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-fault-tolerance</artifactId>
</dependency>
```

D.26.4. MicroProfile Health

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-health</artifactId>
</dependency>
```

Configuration

thorntail.microprofile.health.security-realm

Security realm configuration

D.26.5. MicroProfile JWT RBAC Auth

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-jwt</artifactId>
</dependency>
```

Configuration

thorntail.microprofile.jwt.claims.groups

Default group name. This property can be used to support the JWT tokens without a 'groups' claim.

thorntail.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

thorntail.microprofile.jwt.enabled

Set this to false to disable the MP JWT authentication mechanism. Defaults to true.

thorntail.microprofile.jwt.path.groups

Path to the claim containing an array of groups, for example: 'realm/groups'. It can be used if a token has no 'groups' claim but has the groups set in a different claim

thorntail.microprofile.jwt.realm

Defines the security domain which should be used for MicroProfile JWT. If no security domain with this name exists, one will be created using sensible defaults. If this option is set, then the @LoginConfig annotation is not needed but if it is present then its realmName property, if set, must have the same value as this option.

thorntail.microprofile.jwt.roles.file

Roles properties file path, ignored if the roles.map property is set

thorntail.microprofile.jwt.roles.map

Roles properties map

thorntail.microprofile.jwt.token.cookie

Cookie name containing a JWT token. This property is ignored unless the 'thorntail.microprofile.jwt.token.header' is set to 'Cookie'

thorntail.microprofile.jwt.token.exp-grace-period

The JWT token expiration grace period in seconds

thorntail.microprofile.jwt.token.header

HTTP header which is expected to contain a JWT token, default value is 'Authorization'

thorntail.microprofile.jwt.token.issued-by

The URI of the JWT token issuer

thorntail.microprofile.jwt.token.jwks-refresh-interval

The interval at which the JWKS URI should be queried for keys (in minutes). It is ignored if the value of either signer-pub-key-location or jwks-uri is not HTTPS URI

thorntail.microprofile.jwt.token.jwks-uri

The JWKS URI from which to load public keys. This property is deprecated, use the 'thorntail.microprofile.jwt.token.signer-pub-key-location' property instead

thorntail.microprofile.jwt.token.signer-pub-key

The public key of the JWT token signer. Can be prefixed 'file:' or 'classpath:' to refer to external assets, but this is deprecated; use 'thorntail.microprofile.jwt.token.signer-pub-key-location' instead

thorntail.microprofile.jwt.token.signer-pub-key-location

Location of the public key of the JWT token signer. By default, or when the 'classpath:' prefix is present, this is a classpath resource. Can be prefixed with 'file:' to refer to an external file. Can also be a HTTPS URL of a JWK Set.

D.26.6. MicroProfile Metrics

This fraction implements the Eclipse MicroProfile Metrics 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> ❶
```

1 war packaging

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-metrics</artifactId>
</dependency>
```

D.26.7. MicroProfile OpenAPI

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-openapi</artifactId>
</dependency>
```

D.26.8. MicroProfile OpenTracing

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-opentracing</artifactId>
</dependency>
```

D.26.9. MicroProfile Rest Client

This fraction implements the Eclipse MicroProfile Rest Client specification.

D.26.9.1. 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.26.3, “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.26.9.2. RestClientProxy

In addition to the MicroProfile Rest Client specification, every Rest Client proxy implements **org.jboss.resteasy.microprofile.client.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.27. 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

thorntail.monitor.security-realm

(not yet documented)

D.28. 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.29. NAMING

Provides support for JNDI.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>naming</artifactId>
</dependency>
```

Configuration

thorntail.naming.bindings.*KEY*.attribute-class

The object factory class name for object factory bindings

thorntail.naming.bindings.*KEY*.binding-type

The type of binding to create, may be simple, lookup, external-context or object-factory

thorntail.naming.bindings.*KEY*.cache

If the external context should be cached

thorntail.naming.bindings.*KEY*.environment

The environment to use on object factory instance retrieval

thorntail.naming.bindings.*KEY*.lookup

The entry to lookup in JNDI for lookup bindings

thorntail.naming.bindings.*KEY*.module

The module to load the object factory from for object factory bindings

thorntail.naming.bindings.*KEY*.type

The type of the value to bind for simple bindings, this must be a primitive type

thorntail.naming.bindings.*KEY*.value

The value to bind for simple bindings

D.30. RX-JAVA

Maven Coordinates

```
<dependency>
```

```
  <groupId>io.thorntail</groupId>
  <artifactId>netflix-rxjava</artifactId>
</dependency>
```

D.31. OPENTRACING

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>opentracing</artifactId>
</dependency>
```

Configuration

thorntail.opentracing.servlet.skipPattern

The servlet skip pattern as a Java compilable Pattern. Optional. Ex.: **/health-check**

D.31.1. OpenTracing TracerResolver

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>opentracing-tracerresolver</artifactId>
</dependency>
```

D.32. 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

thorntail.remoting.auth-realm

The authentication realm to use if no authentication CallbackHandler is specified.

thorntail.remoting.authentication-retries

Specify the number of times a client is allowed to retry authentication before closing the connection.

thorntail.remoting.authorize-id

The SASL authorization ID. Used as authentication user name to use if no authentication CallbackHandler is specified and the selected SASL mechanism demands a user name.

thorntail.remoting.buffer-region-size

The size of allocated buffer regions.

thorntail.remoting.connectors.*KEY*.authentication-provider

The "authentication-provider" element contains the name of the authentication provider to use for incoming connections.

thorntail.remoting.connectors.*KEY*.properties.*KEY*.value

The property value.

thorntail.remoting.connectors.*KEY*.sasl-authentication-factory

Reference to the SASL authentication factory to secure this connector.

thorntail.remoting.connectors.*KEY*.sasl-protocol

The protocol to pass into the SASL mechanisms used for authentication.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.remoting.connectors.*KEY*.sasl-security.properties.*KEY*.value

The property value.

thorntail.remoting.connectors.*KEY*.sasl-security.qop

The optional nested "qop" element contains a list of quality-of-protection values, in decreasing order of preference.

thorntail.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.

thorntail.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.

thorntail.remoting.connectors.*KEY*.sasl-security.strength

The optional nested "strength" element contains a list of cipher strength values, in decreasing order of preference.

thorntail.remoting.connectors.*KEY*.security-realm

The associated security realm to use for authentication for this connector.

thorntail.remoting.connectors.*KEY*.server-name

The server name to send in the initial message exchange and for SASL based authentication.

thorntail.remoting.connectors.*KEY*.socket-binding

The name of the socket binding to attach to.

thorntail.remoting.connectors.*KEY*.ssl-context

Reference to the SSLContext to use for this connector.

thorntail.remoting.endpoint-configuration.auth-realm

The authentication realm to use if no authentication CallbackHandler is specified.

thorntail.remoting.endpoint-configuration.authentication-retries

Specify the number of times a client is allowed to retry authentication before closing the connection.

thorntail.remoting.endpoint-configuration.authorize-id

The SASL authorization ID. Used as authentication user name to use if no authentication CallbackHandler is specified and the selected SASL mechanism demands a user name.

thorntail.remoting.endpoint-configuration.buffer-region-size

The size of allocated buffer regions.

thorntail.remoting.endpoint-configuration.heartbeat-interval

The interval to use for connection heartbeat, in milliseconds. If the connection is idle in the outbound direction for this amount of time, a ping message will be sent, which will trigger a corresponding reply message.

thorntail.remoting.endpoint-configuration.max-inbound-channels

The maximum number of inbound channels to support for a connection.

thorntail.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.

thorntail.remoting.endpoint-configuration.max-inbound-messages

The maximum number of concurrent inbound messages on a channel.

thorntail.remoting.endpoint-configuration.max-outbound-channels

The maximum number of outbound channels to support for a connection.

thorntail.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.

thorntail.remoting.endpoint-configuration.max-outbound-messages

The maximum number of concurrent outbound messages on a channel.

thorntail.remoting.endpoint-configuration.receive-buffer-size

The size of the largest buffer that this endpoint will accept over a connection.

thorntail.remoting.endpoint-configuration.receive-window-size

The maximum window size of the receive direction for connection channels, in bytes.

thorntail.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.

thorntail.remoting.endpoint-configuration.send-buffer-size

The size of the largest buffer that this endpoint will transmit over a connection.

thorntail.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.

thorntail.remoting.endpoint-configuration.transmit-window-size

The maximum window size of the transmit direction for connection channels, in bytes.

thorntail.remoting.endpoint-configuration.worker

Worker to use

thorntail.remoting.heartbeat-interval

The interval to use for connection heartbeat, in milliseconds. If the connection is idle in the outbound direction for this amount of time, a ping message will be sent, which will trigger a corresponding reply message.

thorntail.remoting.http-connectors.*KEY*.authentication-provider

The "authentication-provider" element contains the name of the authentication provider to use for incoming connections.

thorntail.remoting.http-connectors.*KEY*.connector-ref

The name (or names) of a connector in the Undertow subsystem to connect to.

thorntail.remoting.http-connectors.*KEY*.properties.*KEY*.value

The property value.

thorntail.remoting.http-connectors.*KEY*.sasl-authentication-factory

Reference to the SASL authentication factory to use for this connector.

thorntail.remoting.http-connectors.*KEY*.sasl-protocol

The protocol to pass into the SASL mechanisms used for authentication.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.remoting.http-connectors.*KEY*.sasl-security.properties.*KEY*.value

The property value.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.remoting.http-connectors.*KEY*.sasl-security.strength

The optional nested "strength" element contains a list of cipher strength values, in decreasing order of preference.

thorntail.remoting.http-connectors.*KEY*.security-realm

The associated security realm to use for authentication for this connector.

thorntail.remoting.http-connectors.*KEY*.server-name

The server name to send in the initial message exchange and for SASL based authentication.

thorntail.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.

thorntail.remoting.local-outbound-connections.*KEY*.properties.*KEY*.value

The property value.

thorntail.remoting.max-inbound-channels

The maximum number of inbound channels to support for a connection.

thorntail.remoting.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.

thorntail.remoting.max-inbound-messages

The maximum number of concurrent inbound messages on a channel.

thorntail.remoting.max-outbound-channels

The maximum number of outbound channels to support for a connection.

thorntail.remoting.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.

thorntail.remoting.max-outbound-messages

The maximum number of concurrent outbound messages on a channel.

thorntail.remoting.outbound-connections.*KEY*.properties.*KEY*.value

The property value.

thorntail.remoting.outbound-connections.*KEY*.uri

The connection URI for the outbound connection.

thorntail.remoting.port

Port for legacy remoting connector

thorntail.remoting.receive-buffer-size

The size of the largest buffer that this endpoint will accept over a connection.

thorntail.remoting.receive-window-size

The maximum window size of the receive direction for connection channels, in bytes.

thorntail.remoting.remote-outbound-connections.*KEY*.authentication-context

Reference to the authentication context instance containing the configuration for outbound connections.

thorntail.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.

thorntail.remoting.remote-outbound-connections.*KEY*.properties.*KEY*.value

The property value.

thorntail.remoting.remote-outbound-connections.*KEY*.protocol

The protocol to use for the remote connection.

thorntail.remoting.remote-outbound-connections.*KEY*.security-realm

Reference to the security realm to use to obtain the password and SSL configuration.

thorntail.remoting.remote-outbound-connections.*KEY*.username

The user name to use when authenticating against the remote server.

thorntail.remoting.required

(not yet documented)

thorntail.remoting.sasl-protocol

Where a SaslServer or SaslClient are created by default the protocol specified it 'remoting', this can be used to override this.

thorntail.remoting.send-buffer-size

The size of the largest buffer that this endpoint will transmit over a connection.

thorntail.remoting.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.

thorntail.remoting.transmit-window-size

The maximum window size of the transmit direction for connection channels, in bytes.

thorntail.remoting.worker

Worker to use

thorntail.remoting.worker-read-threads

The number of read threads to create for the remoting worker.

thorntail.remoting.worker-task-core-threads

The number of core threads for the remoting worker task thread pool.

thorntail.remoting.worker-task-keepalive

The number of milliseconds to keep non-core remoting worker task threads alive.

thorntail.remoting.worker-task-limit

The maximum number of remoting worker tasks to allow before rejecting.

thorntail.remoting.worker-task-max-threads

The maximum number of threads for the remoting worker task thread pool.

thorntail.remoting.worker-write-threads

The number of write threads to create for the remoting worker.

D.33. 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

thorntail.request-controller.active-requests

The number of requests that are currently running in the server

thorntail.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.

thorntail.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.34. RESOURCE ADAPTERS

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>resource-adapters</artifactId>
</dependency>
```

Configuration

thorntail.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.class-name

Specifies the fully qualified class name of an administration object.

thorntail.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.config-properties.*KEY*.value

Custom defined config property value.

thorntail.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.enabled

Specifies if the administration object should be enabled.

thorntail.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.jndi-name

Specifies the JNDI name for the administration object.

thorntail.resource-adapters.resource-adapters.*KEY*.admin-objects.*KEY*.use-java-context

Setting this to false will bind the object into global JNDI.

thorntail.resource-adapters.resource-adapters.*KEY*.archive

Specifies the resource adapter archive.

thorntail.resource-adapters.resource-adapters.*KEY*.beanvalidationgroups

Specifies the bean validation groups that should be used.

thorntail.resource-adapters.resource-adapters.*KEY*.bootstrap-context

Specifies the unique name of the bootstrap context that should be used.

thorntail.resource-adapters.resource-adapters.*KEY*.config-properties.*KEY*.value

Custom defined config property value.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.capacity-decrementer-class

Class defining the policy for decrementing connections in the pool.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.capacity-incrementer-class

Class defining the policy for incrementing connections in the pool.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.class-name

Specifies the fully qualified class name of a managed connection factory or admin object.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.config-properties.*KEY*.value

Custom defined config property value.

thorntail.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.

thorntail.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).

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.enabled

Specifies if the resource adapter should be enabled.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.enlistment

Defines if lazy enlistment should be used if supported by the resource adapter.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.enlistment-trace

Defines if WildFly/IronJacamar should record enlistment traces.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.flush-strategy

Specifies how the pool should be flushed in case of an error.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.initial-pool-size

Specifies the initial number of connections a pool should hold.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.interleaving

An element to enable interleaving for XA connections.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.jndi-name

Specifies the JNDI name for the connection factory.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.mcp

Defines the ManagedConnectionPool implementation. For example: org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.min-pool-size

Specifies the minimum number of connections for a pool.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.no-recovery

Specifies if the connection pool should be excluded from recovery.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.pad-xid

Specifies whether the Xid should be padded.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.pool-fair

Defines if pool use should be fair.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.pool-prefill

Specifies if the pool should be prefilled. Changing this value requires a server restart.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.pool-use-strict-min

Specifies if the min-pool-size should be considered strict.

thorntail.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).

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-credential-reference

Credential (from Credential Store) to authenticate on recovery connection

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-elytron-enabled

Indicates that an Elytron authentication context will be used for recovery.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-password

The password used for recovery.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-plugin-class-name

The fully qualified class name of the recovery plugin implementation.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-plugin-properties

The properties for the recovery plugin.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-security-domain

The PicketBox security domain used for recovery.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.recovery-username

The user name used for recovery.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.sharable

Enable the use of sharable connections, which allows lazy association to be enabled if supported.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.tracking

Defines if IronJacamar should track connection handles across transaction boundaries.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.use-ccm

Enable the use of a cached connection manager.

thorntail.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).

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.use-java-context

Setting this to false will bind the object into global JNDI.

thorntail.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.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.connection-definitions.*KEY*.xa-resource-timeout

The value is passed to XAResource.setTransactionTimeout(), in seconds.

thorntail.resource-adapters.resource-adapters.*KEY*.module

Specifies the module from which resource adapter will be loaded

thorntail.resource-adapters.resource-adapters.*KEY*.statistics-enabled

Define whether runtime statistics are enabled or not.

thorntail.resource-adapters.resource-adapters.*KEY*.transaction-support

Specifies the transaction support level of the resource adapter.

thorntail.resource-adapters.resource-adapters.*KEY*.wm-elytron-security-domain

Defines the name of the Elytron security domain that should be used.

thorntail.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.

thorntail.resource-adapters.resource-adapters.*KEY*.wm-security-default-groups

Defines a default groups list that should be added to the used Subject instance.

thorntail.resource-adapters.resource-adapters.*KEY*.wm-security-default-principal

Defines a default principal name that should be added to the used Subject instance.

thorntail.resource-adapters.resource-adapters.*KEY*.wm-security-domain

Defines the name of the PicketBox security domain that should be used.

thorntail.resource-adapters.resource-adapters.*KEY*.wm-security-mapping-groups

List of groups mappings.

thorntail.resource-adapters.resource-adapters.*KEY*.wm-security-mapping-required

Defines if a mapping is required for security credentials.

thorntail.resource-adapters.resource-adapters.*KEY*.wm-security-mapping-users

List of user mappings.

D.35. SECURITY

Provides underlying security infrastructure to support JAAS and other security APIs.

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>security</artifactId>
</dependency>
```

Configuration

thorntail.security.classic-vault.code

Fully Qualified Name of the Security Vault Implementation.

thorntail.security.classic-vault.vault-options

Security Vault options.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.security.elytron-realms.*KEY*.legacy-jaas-config

The name of the legacy security domain to which authentication will be delegated.

thorntail.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.

thorntail.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.

thorntail.security.initialize-jacc

Indicates if this subsystem should be in charge of initializing JACC related services.

thorntail.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.

thorntail.security.security-domains.*KEY*.classic-acl.acl-modules.*KEY*.code

Class name of the module to be instantiated.

thorntail.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.

thorntail.security.security-domains.*KEY*.classic-acl.acl-modules.*KEY*.module

Name of JBoss Module where the login module is located.

thorntail.security.security-domains.*KEY*.classic-acl.acl-modules.*KEY*.module-options

List of module options containing a name/value pair.

thorntail.security.security-domains.*KEY*.classic-audit.provider-modules.*KEY*.code

Class name of the module to be instantiated.

thorntail.security.security-domains.*KEY*.classic-audit.provider-modules.*KEY*.module

Name of JBoss Module where the mapping module code is located.

thorntail.security.security-domains.*KEY*.classic-audit.provider-modules.*KEY*.module-options

List of module options containing a name/value pair.

thorntail.security.security-domains.*KEY*.classic-authentication.login-modules.*KEY*.code

Class name of the module to be instantiated.

thorntail.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.

thorntail.security.security-domains.*KEY*.classic-authentication.login-modules.*KEY*.module

Name of JBoss Module where the login module is located.

thorntail.security.security-domains.*KEY*.classic-authentication.login-modules.*KEY*.module-options

List of module options containing a name/value pair.

thorntail.security.security-domains.*KEY*.classic-authorization.policy-modules.*KEY*.code

Class name of the module to be instantiated.

thorntail.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.

thorntail.security.security-domains.*KEY*.classic-authorization.policy-modules.*KEY*.module

Name of JBoss Module where the login module is located.

thorntail.security.security-domains.*KEY*.classic-authorization.policy-modules.*KEY*.module-options

List of module options containing a name/value pair.

thorntail.security.security-domains.*KEY*.classic-identity-trust.trust-modules.*KEY*.code

Class name of the module to be instantiated.

thorntail.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.

thorntail.security.security-domains.*KEY*.classic-identity-trust.trust-modules.*KEY*.module

Name of JBoss Module where the login module is located.

thorntail.security.security-domains.*KEY*.classic-identity-trust.trust-modules.*KEY*.module-options

List of module options containing a name/value pair.

thorntail.security.security-domains.*KEY*.classic-jsse.additional-properties

Additional properties that may be necessary to configure JSSE.

thorntail.security.security-domains.*KEY*.classic-jsse.cipher-suites

Comma separated list of cipher suites to enable on SSLSockets.

thorntail.security.security-domains.*KEY*.classic-jsse.client-alias

Preferred alias to use when the KeyManager chooses the client alias.

thorntail.security.security-domains.*KEY*.classic-jsse.client-auth

Boolean attribute to indicate if client’s certificates should also be authenticated on the server side.

thorntail.security.security-domains.*KEY*.classic-jsse.key-manager

JSEE Key Manager factory

thorntail.security.security-domains.*KEY*.classic-jsse.keystore

Configures a JSSE key store

thorntail.security.security-domains.*KEY*.classic-jsse.protocols

Comma separated list of protocols to enable on SSLSockets.

thorntail.security.security-domains.*KEY*.classic-jsse.server-alias

Preferred alias to use when the KeyManager chooses the server alias.

thorntail.security.security-domains.*KEY*.classic-jsse.service-auth-token

Token to retrieve PrivateKeys from the KeyStore.

thorntail.security.security-domains.*KEY*.classic-jsse.trust-manager

JSEE Trust Manager factory

thorntail.security.security-domains.*KEY*.classic-jsse.truststore

Configures a JSSE trust store

thorntail.security.security-domains.*KEY*.classic-mapping.mapping-modules.*KEY*.code

Class name of the module to be instantiated.

thorntail.security.security-domains.*KEY*.classic-mapping.mapping-modules.*KEY*.module

Name of JBoss Module where the mapping module code is located.

thorntail.security.security-domains.*KEY*.classic-mapping.mapping-modules.*KEY*.module-options

List of module options containing a name/value pair.

thorntail.security.security-domains.*KEY*.classic-mapping.mapping-modules.*KEY*.type

Type of mapping this module performs. Allowed values are principal, role, attribute or credential..

thorntail.security.security-domains.*KEY*.jaspi-authentication.auth-modules.*KEY*.code

Class name of the module to be instantiated.

thorntail.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.

thorntail.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.

thorntail.security.security-domains.*KEY*.jaspi-authentication.auth-modules.*KEY*.module

Name of JBoss Module where the mapping module code is located.

thorntail.security.security-domains.*KEY*.jaspi-authentication.auth-modules.*KEY*.module-options

List of module options containing a name/value pair.

thorntail.security.security-domains.*KEY*.jaspi-authentication.login-module-stacks.*KEY*.login-modules.*KEY*.code

Class name of the module to be instantiated.

thorntail.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.

thorntail.security.security-domains.*KEY*.jaspi-authentication.login-module-stacks.*KEY*.login-modules.*KEY*.module

Name of JBoss Module where the login module is located.

thorntail.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.36. TOPOLOGY

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>topology</artifactId>
</dependency>
```

D.36.1. OpenShift

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>topology-openshift</artifactId>
</dependency>
```

D.36.2. Topology UI

Maven Coordinates

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>topology-webapp</artifactId>
</dependency>
```

Configuration

thorntail.topology.web-app.expose-topology-endpoint

Flag to enable or disable the topology web endpoint

thorntail.topology.web-app.proxied-service-mappings

Service name to URL path proxy mappings

D.37. 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

thorntail.transactions.average-commit-time

The average time of transaction commit, measured from the moment the client calls commit until the transaction manager determines that the commit attempt was successful.

thorntail.transactions.commit-markable-resources.*KEY*.batch-size

Batch size for this CMR resource

thorntail.transactions.commit-markable-resources.*KEY*.immediate-cleanup

Immediate cleanup associated to this CMR resource

thorntail.transactions.commit-markable-resources.*KEY*.jndi-name

JNDi name of this CMR resource

thorntail.transactions.commit-markable-resources.*KEY*.name

table name for storing XIDs

thorntail.transactions.default-timeout

The default timeout for a transaction managed by the transaction manager.

thorntail.transactions.enable-statistics

Whether transaction statistics should be gathered.

thorntail.transactions.enable-tsm-status

Whether the transaction status manager (TSM) service, needed for out of process recovery, should be provided or not.

thorntail.transactions.hornetq-store-enable-async-io

Whether AsyncIO should be enabled for the journal store.

thorntail.transactions.jdbc-action-store-drop-table

Configure if jdbc action store should drop tables.

thorntail.transactions.jdbc-action-store-table-prefix

Optional prefix for table used to write transaction logs in configured jdbc action store.

thorntail.transactions.jdbc-communication-store-drop-table

Configure if jdbc communication store should drop tables.

thorntail.transactions.jdbc-communication-store-table-prefix

Optional prefix for table used to write transaction logs in configured jdbc communication store.

thorntail.transactions.jdbc-state-store-drop-table

Configure if jdbc state store should drop tables.

thorntail.transactions.jdbc-state-store-table-prefix

Optional prefix for table used to write transaction logs in configured jdbc state store.

thorntail.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".

thorntail.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.

thorntail.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.

thorntail.transactions.log-store.expose-all-logs

Whether to expose all logs like orphans etc. By default only a subset of transaction logs is exposed.

thorntail.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.

thorntail.transactions.log-store.transactions.*KEY*.id

The id of this transaction.

thorntail.transactions.log-store.transactions.*KEY*.jmx-name

The JMX name of this transaction.

thorntail.transactions.log-store.transactions.*KEY*.participants.*KEY*.eis-product-name

The JCA enterprise information system’s product name.

thorntail.transactions.log-store.transactions.*KEY*.participants.*KEY*.eis-product-version

The JCA enterprise information system’s product version

thorntail.transactions.log-store.transactions.*KEY*.participants.*KEY*.jmx-name

The JMX name of this participant.

thorntail.transactions.log-store.transactions.*KEY*.participants.*KEY*.jndi-name

JNDI name of this participant.

thorntail.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).

thorntail.transactions.log-store.transactions.*KEY*.participants.*KEY*.type

The type name under which this record is stored.

thorntail.transactions.log-store.transactions.*KEY*.type

The type name under which this record is stored.

thorntail.transactions.log-store.type

Specifies the implementation type of the logging store.

thorntail.transactions.maximum-timeout

If the default timeout is zero then this value is consulted to set the maximum timeout (in seconds) for a transaction managed by the transaction manager.

thorntail.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.

thorntail.transactions.number-of-aborted-transactions

The number of aborted (i.e. rolledback) transactions.

thorntail.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.

thorntail.transactions.number-of-committed-transactions

The number of committed transactions.

thorntail.transactions.number-of-heuristics

The number of transactions which have terminated with heuristic outcomes.

thorntail.transactions.number-of-inflight-transactions

The number of transactions that have begun but not yet terminated.

thorntail.transactions.number-of-nested-transactions

The total number of nested (sub) transactions created.

thorntail.transactions.number-of-resource-rollbacks

The number of transactions that rolled back due to resource (participant) failure.

thorntail.transactions.number-of-system-rollbacks

The number of transactions that have been rolled back due to internal system errors.

thorntail.transactions.number-of-timed-out-transactions

The number of transactions that have rolled back due to timeout.

thorntail.transactions.number-of-transactions

The total number of transactions (top-level and nested) created

thorntail.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.

thorntail.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.

thorntail.transactions.port

Port for transaction manager

thorntail.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.

thorntail.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'.

thorntail.transactions.process-id-uuid

Indicates whether the transaction manager should use a UUID based process id.

thorntail.transactions.recovery-listener

Used to specify if the recovery system should listen on a network socket or not.

thorntail.transactions.socket-binding

Used to reference the correct socket binding to use for the recovery environment.

thorntail.transactions.statistics-enabled

Whether transaction statistics should be gathered.

thorntail.transactions.status-port

Status port for transaction manager

thorntail.transactions.status-socket-binding

Used to reference the correct socket binding to use for the transaction status manager.

thorntail.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.

thorntail.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.

thorntail.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.38. 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

thorntail.ajp.enable

Determine if AJP should be enabled

thorntail.ajp.port

Set the port for the default AJP listener

thorntail.deployment

Map of security configuration by deployment

thorntail.http.port

Set the port for the default HTTP listener

thorntail.https.certificate.generate

Should a self-signed certificate be generated

thorntail.https.certificate.generate.host

Hostname for the generated self-signed certificate

thorntail.https.key.alias

Alias to the server certificate key entry in the keystore

thorntail.https.key.password

Password to the server certificate

thorntail.https.keystore.embedded

Should an embedded keystore be created

thorntail.https.keystore.password

Password to the server keystore

thorntail.https.keystore.path

Path to the server keystore

thorntail.https.only

Only enable the HTTPS Listener

thorntail.https.port

Set the port for the default HTTPS listener

thorntail.undertow.application-security-domains.*KEY*.enable-jacc

Enable authorization using JACC

thorntail.undertow.application-security-domains.*KEY*.enable-jaspi

Enable JASPI authentication for the associated deployments.

thorntail.undertow.application-security-domains.*KEY*.http-authentication-factory

The HTTP Authentication Factory to be used by deployments that reference the mapped security domain.

thorntail.undertow.application-security-domains.*KEY*.integrated-jaspi

Whether integrated JASPI should be used. If 'false', an ad hoc identity will be created instead.

thorntail.undertow.application-security-domains.*KEY*.override-deployment-config

Should the authentication configuration in the deployment be overridden by the factory.

thorntail.undertow.application-security-domains.*KEY*.referencing-deployments

The deployments currently referencing this mapping.

thorntail.undertow.application-security-domains.*KEY*.security-domain

The SecurityDomain to be used by deployments that reference the mapped security domain.

thorntail.undertow.application-security-domains.*KEY*.single-sign-on-setting.client-ssl-context

Reference to the SSL context used to secure back-channel logout connection.

thorntail.undertow.application-security-domains.*KEY*.single-sign-on-setting.cookie-name

Name of the cookie

thorntail.undertow.application-security-domains.*KEY*.single-sign-on-setting.credential-reference

The credential reference to decrypt the private key entry.

thorntail.undertow.application-security-domains.*KEY*.single-sign-on-setting.domain

The cookie domain that will be used.

thorntail.undertow.application-security-domains.*KEY*.single-sign-on-setting.http-only

Set Cookie httpOnly attribute.

thorntail.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.

thorntail.undertow.application-security-domains.*KEY*.single-sign-on-setting.key-store

Reference to key store containing a private key entry.

thorntail.undertow.application-security-domains.*KEY*.single-sign-on-setting.path

Cookie path.

thorntail.undertow.application-security-domains.*KEY*.single-sign-on-setting.secure

Set Cookie secure attribute.

thorntail.undertow.buffer-caches.*KEY*.buffer-size

The size of an individual buffer, in bytes.

thorntail.undertow.buffer-caches.*KEY*.buffers-per-region

The numbers of buffers in a region

thorntail.undertow.buffer-caches.*KEY*.max-regions

The maximum number of regions

thorntail.undertow.byte-buffer-pools.*KEY*.buffer-size

The size of the buffer

thorntail.undertow.byte-buffer-pools.*KEY*.direct

If this is true the buffer pool will use direct buffers, this is recommended for best performance

thorntail.undertow.byte-buffer-pools.*KEY*.leak-detection-percent

The percentage of buffers that will be allocated with a leak detector. This should only be larger than zero if you are experiencing issues with buffers leaking.

thorntail.undertow.byte-buffer-pools.*KEY*.max-pool-size

The maximum amount of buffers to keep in the pool. If more buffers are required at runtime they will be allocated dynamically. Setting this to zero effectively disables pooling.

thorntail.undertow.byte-buffer-pools.*KEY*.thread-local-cache-size

The maximum number of buffers to cache on each thread. The actual number may be lower depending on the calculated usage pattern.

thorntail.undertow.default-security-domain

The default security domain used by web deployments

thorntail.undertow.default-server

The default server to use for deployments

thorntail.undertow.default-servlet-container

The default servlet container to use for deployments

thorntail.undertow.default-virtual-host

The default virtual host to use for deployments

thorntail.undertow.filter-configuration.custom-filters.*KEY*.class-name

Class name of HttpHandler

thorntail.undertow.filter-configuration.custom-filters.*KEY*.module

Module name where class can be loaded from

thorntail.undertow.filter-configuration.custom-filters.*KEY*.parameters

Filter parameters

thorntail.undertow.filter-configuration.error-pages.*KEY*.code

Error page code

thorntail.undertow.filter-configuration.error-pages.*KEY*.path

Error page path

thorntail.undertow.filter-configuration.expression-filters.*KEY*.expression

The expression that defines the filter

thorntail.undertow.filter-configuration.expression-filters.*KEY*.module

Module to use to load the filter definitions

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.advertise-frequency

The frequency (in milliseconds) that mod-cluster advertises itself on the network

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.advertise-path

The path that mod-cluster is registered under.

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.advertise-protocol

The protocol that is in use.

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.advertise-socket-binding

The multicast group and port that is used to advertise.

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.max-attempts

Maximum number of failover attempts by reverse proxy when sending the request to the backend server.

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.aliases

The nodes aliases

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.cache-connections

The number of connections to keep alive indefinitely

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.contexts.*KEY*.requests

The number of requests against this context

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.contexts.*KEY*.status

The status of this context

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.elected

The elected count

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.flush-packets

If received data should be immediately flushed

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.load

The current load of this node

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.load-balancing-group

The load balancing group this node belongs to

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.max-connections

The maximum number of connections per IO thread

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.open-connections

The current number of open connections

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.ping

The nodes ping

thorntail.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

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.read

The number of bytes read from the node

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.request-queue-size

The size of the request queue

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.status

The current status of this node

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.timeout

The request timeout

thorntail.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

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.uri

The URI that the load balancer uses to connect to the node

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.nodes.*KEY*.written

The number of bytes transferred to the node

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.sticky-session

If sticky sessions are enabled

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.sticky-session-cookie

The session cookie name

thorntail.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

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.sticky-session-path

The path of the sticky session cookie

thorntail.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

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.balancers.*KEY*.wait-worker

The number of seconds to wait for an available worker

thorntail.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

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.cached-connections-per-thread

The number of connections that will be kept alive indefinitely

thorntail.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)

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.connections-per-thread

The number of connections that will be maintained to backend servers, per IO thread.

thorntail.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

thorntail.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.

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.health-check-interval

The frequency of health check pings to backend nodes

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.http2-enable-push

If push should be enabled for HTTP/2 connections

thorntail.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.

thorntail.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

thorntail.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

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.http2-max-frame-size

The max HTTP/2 frame size

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.http2-max-header-list-size

The maximum size of request headers the server is prepared to accept

thorntail.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

thorntail.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.

thorntail.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.

thorntail.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

thorntail.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).

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.ranked-affinity.delimiter

The delimiter used to separate ranked routes within the session ID.

thorntail.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

thorntail.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.

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.security-realm

The security realm that provides the SSL configuration

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.ssl-context

Reference to the SSLContext to be used by this filter.

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.use-alias

If an alias check is performed

thorntail.undertow.filter-configuration.mod-clusters.*KEY*.worker

The XNIO worker that is used to send the advertise notifications

thorntail.undertow.filter-configuration.request-limits.*KEY*.max-concurrent-requests

Maximum number of concurrent requests

thorntail.undertow.filter-configuration.request-limits.*KEY*.queue-size

Number of requests to queue before they start being rejected

thorntail.undertow.filter-configuration.response-headers.*KEY*.header-name

Header name

thorntail.undertow.filter-configuration.response-headers.*KEY*.header-value

Value for header

thorntail.undertow.filter-configuration.rewrites.*KEY*.redirect

If this is true then a redirect will be done instead of a rewrite

thorntail.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

thorntail.undertow.handler-configuration.files.*KEY*.cache-buffer-size

Size of the buffers, in bytes.

thorntail.undertow.handler-configuration.files.*KEY*.cache-buffers

Number of buffers

thorntail.undertow.handler-configuration.files.*KEY*.case-sensitive

Use case sensitive file handling

thorntail.undertow.handler-configuration.files.*KEY*.directory-listing

Enable directory listing?

thorntail.undertow.handler-configuration.files.*KEY*.follow-symlink

Enable following symbolic links

thorntail.undertow.handler-configuration.files.*KEY*.path

Path on filesystem from where file handler will serve resources

thorntail.undertow.handler-configuration.files.*KEY*.safe-symlink-paths

Paths that are safe to be targets of symbolic links

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.cached-connections-per-thread

The number of connections that will be kept alive indefinitely

thorntail.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)

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.connections-per-thread

The number of connections that will be maintained to backend servers, per IO thread.

thorntail.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.

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.instance-id

The instance id (aka JVM route) that will be used to enable sticky sessions

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.outbound-socket-binding

Outbound socket binding for this host

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.path

Optional path if host is using non root resource

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.scheme

What kind of scheme is used

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.security-realm

The security realm that provides the SSL configuration for the connection to the host

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.hosts.*KEY*.ssl-context

Reference to the SSLContext to be used by this handler.

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.max-request-time

The maximum time that a proxy request can be active for, before being killed

thorntail.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).

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.problem-server-retry

Time in seconds to wait before attempting to reconnect to a server that is down

thorntail.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

thorntail.undertow.handler-configuration.reverse-proxies.*KEY*.session-cookie-names

Comma separated list of session cookie names. Generally this will just be JSESSIONID.

thorntail.undertow.instance-id

The cluster instance id

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.allow-encoded-slash

If a request comes in with encoded / characters (i.e. %2F), will these be decoded.

thorntail.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.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.allow-unescaped-characters-in-url

If this is true Undertow will accept non-encoded characters that are disallowed by the URI specification. This defaults to false, and in general should not be needed as most clients correctly encode characters. Note that setting this to true can be considered a security risk, as allowing non-standard characters can allow request smuggling attacks in some circumstances.

thorntail.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.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.buffer-pipelined-data

If we should buffer pipelined requests.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.buffer-pool

The listeners buffer pool

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.bytes-received

The number of bytes that have been received by this listener

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.bytes-sent

The number of bytes that have been sent out on this listener

thorntail.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.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.disallowed-methods

A comma separated list of HTTP methods that are not allowed

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.enabled

If the listener is enabled

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.error-count

The number of 500 responses that have been sent by this listener

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-header-size

The maximum size of a http request header, in bytes.

thorntail.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.

thorntail.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).

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-post-size

The maximum size of a post that will be accepted, in bytes.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.max-processing-time

The maximum processing time taken by a request on this listener

thorntail.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.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.processing-time

The total processing time of all requests handed by this listener

thorntail.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}.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.receive-buffer

The receive buffer size, in bytes.

thorntail.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

thorntail.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.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.request-count

The number of requests this listener has served

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.request-parse-timeout

The maximum amount of time (in milliseconds) that can be spent parsing the request

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.resolve-peer-address

Enables host dns lookup

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.rfc6265-cookie-validation

If cookies should be validated to ensure they comply with RFC6265.

thorntail.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

The listener scheme, can be HTTP or HTTPS. By default the scheme will be taken from the incoming AJP request.

thorntail.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.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.send-buffer

The send buffer size, in bytes.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.socket-binding

The listener socket binding

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.tcp-backlog

Configure a server with the specified backlog.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.tcp-keep-alive

Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.url-charset

URL charset

thorntail.undertow.servers.*KEY*.ajp-listeners.*KEY*.worker

The listeners XNIO worker

thorntail.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}.

thorntail.undertow.servers.*KEY*.default-host

The servers default virtual host

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.directory

Directory in which to save logs

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.extended

If the log uses the extended log file format

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.pattern

The access log pattern.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.predicate

Predicate that determines if the request should be logged

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.prefix

Prefix for the log file name.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.relative-to

The directory the path is relative to

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.rotate

Rotate the access log every day.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.suffix

Suffix for the log file name.

thorntail.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.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.access-log-setting.worker

Name of the worker to use for logging

thorntail.undertow.servers.*KEY*.hosts.*KEY*.alias

Aliases for the host

thorntail.undertow.servers.*KEY*.hosts.*KEY*.console-access-log-setting.attributes

The attributes to be included in the structured output.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.console-access-log-setting.include-host-name

Indicates whether or not the host name should included in the JSON structured output. If set to true the key will be hostName in the structured data and the value will be the host this console-access-log belongs to.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.console-access-log-setting.metadata

Any additional metadata to add to the JSON structured output.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.console-access-log-setting.predicate

Predicate that determines if the request should be logged.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.console-access-log-setting.worker

Name of the worker to use for logging.

thorntail.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.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.default-web-module

Default web module

thorntail.undertow.servers.*KEY*.hosts.*KEY*.disable-console-redirect

if set to true, /console redirect wont be enabled for this host, default is false

thorntail.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.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.filter-refs.*KEY*.priority

Defines filter order. A lower number instructs the server to be included earlier in the handler chain than others with higher numbers. Values range from 1, indicating the filter will be handled first, to 2147483647, resulting in the filter being handled last.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.http-invoker-setting.http-authentication-factory

The HTTP authentication factory to use for authentication

thorntail.undertow.servers.*KEY*.hosts.*KEY*.http-invoker-setting.path

The path that the services are installed under

thorntail.undertow.servers.*KEY*.hosts.*KEY*.http-invoker-setting.security-realm

The legacy security realm to use for authentication

thorntail.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.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.locations.*KEY*.filter-refs.*KEY*.priority

Defines filter order. A lower number instructs the server to be included earlier in the handler chain than others with higher numbers. Values range from 1, indicating the filter will be handled first, to 2147483647, resulting in the filter being handled last.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.locations.*KEY*.handler

Default handler for this location

thorntail.undertow.servers.*KEY*.hosts.*KEY*.queue-requests-on-start

If requests should be queued on start for this host. If this is set to false the default response code will be returned instead.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.cookie-name

Name of the cookie

thorntail.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.domain

The cookie domain that will be used.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.http-only

Set Cookie httpOnly attribute.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.path

Cookie path.

thorntail.undertow.servers.*KEY*.hosts.*KEY*.single-sign-on-setting.secure

Set Cookie secure attribute.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.allow-encoded-slash

If a request comes in with encoded / characters (i.e. %2F), will these be decoded.

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.allow-unescaped-characters-in-url

If this is true Undertow will accept non-encoded characters that are disallowed by the URI specification. This defaults to false, and in general should not be needed as most clients correctly encode characters. Note that setting this to true can be considered a security risk, as allowing non-standard characters can allow request smuggling attacks in some circumstances.

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.buffer-pipelined-data

If we should buffer pipelined requests.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.buffer-pool

The listeners buffer pool

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.bytes-received

The number of bytes that have been received by this listener

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.bytes-sent

The number of bytes that have been sent out on this listener

thorntail.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.

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.disallowed-methods

A comma separated list of HTTP methods that are not allowed

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.enable-http2

Enables HTTP2 support for this listener

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.enabled

If the listener is enabled

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.error-count

The number of 500 responses that have been sent by this listener

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.http2-enable-push

If server push is enabled for this connection

thorntail.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.

thorntail.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

thorntail.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

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.http2-max-frame-size

The max HTTP/2 frame size

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.http2-max-header-list-size

The maximum size of request headers the server is prepared to accept

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.max-header-size

The maximum size of a http request header, in bytes.

thorntail.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.

thorntail.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).

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.max-post-size

The maximum size of a post that will be accepted, in bytes.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.max-processing-time

The maximum processing time taken by a request on this listener

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.processing-time

The total processing time of all requests handed by this listener

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.proxy-protocol

If this is true then the listener will use the proxy protocol v1, as defined by https://www.haproxy.org/download/1.8/doc/proxy-protocol.txt. This option MUST only be enabled for listeners that are behind a load balancer that supports the same protocol.

thorntail.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}.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.receive-buffer

The receive buffer size, in bytes.

thorntail.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

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.request-count

The number of requests this listener has served

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.request-parse-timeout

The maximum amount of time (in milliseconds) that can be spent parsing the request

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.resolve-peer-address

Enables host dns lookup

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.rfc6265-cookie-validation

If cookies should be validated to ensure they comply with RFC6265.

thorntail.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.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.send-buffer

The send buffer size, in bytes.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.socket-binding

The listener socket binding

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.tcp-backlog

Configure a server with the specified backlog.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.tcp-keep-alive

Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.url-charset

URL charset

thorntail.undertow.servers.*KEY*.http-listeners.*KEY*.worker

The listeners XNIO worker

thorntail.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}.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.allow-encoded-slash

If a request comes in with encoded / characters (i.e. %2F), will these be decoded.

thorntail.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.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.allow-unescaped-characters-in-url

If this is true Undertow will accept non-encoded characters that are disallowed by the URI specification. This defaults to false, and in general should not be needed as most clients correctly encode characters. Note that setting this to true can be considered a security risk, as allowing non-standard characters can allow request smuggling attacks in some circumstances.

thorntail.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.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.buffer-pipelined-data

If we should buffer pipelined requests.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.buffer-pool

The listeners buffer pool

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.bytes-received

The number of bytes that have been received by this listener

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.bytes-sent

The number of bytes that have been sent out on this listener

thorntail.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.

thorntail.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.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.disallowed-methods

A comma separated list of HTTP methods that are not allowed

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.enable-http2

Enables HTTP2 support for this listener

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.enable-spdy

Enables SPDY support for this listener. This has been deprecated and has no effect, HTTP/2 should be used instead

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.enabled

If the listener is enabled

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.enabled-cipher-suites

Configures Enabled SSL ciphers

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.enabled-protocols

Configures SSL protocols

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.error-count

The number of 500 responses that have been sent by this listener

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.http2-enable-push

If server push is enabled for this connection

thorntail.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.

thorntail.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

thorntail.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

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.http2-max-frame-size

The max HTTP/2 frame size

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.http2-max-header-list-size

The maximum size of request headers the server is prepared to accept

thorntail.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.

thorntail.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.

thorntail.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.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.max-header-size

The maximum size of a http request header, in bytes.

thorntail.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.

thorntail.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).

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.max-post-size

The maximum size of a post that will be accepted, in bytes.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.max-processing-time

The maximum processing time taken by a request on this listener

thorntail.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.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.processing-time

The total processing time of all requests handed by this listener

thorntail.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.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.proxy-protocol

If this is true then the listener will use the proxy protocol v1, as defined by https://www.haproxy.org/download/1.8/doc/proxy-protocol.txt. This option MUST only be enabled for listeners that are behind a load balancer that supports the same protocol.

thorntail.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}.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.receive-buffer

The receive buffer size, in bytes.

thorntail.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

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.request-count

The number of requests this listener has served

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.request-parse-timeout

The maximum amount of time (in milliseconds) that can be spent parsing the request

thorntail.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.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.resolve-peer-address

Enables host dns lookup

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.rfc6265-cookie-validation

If cookies should be validated to ensure they comply with RFC6265.

thorntail.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.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.security-realm

The listeners security realm

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.send-buffer

The send buffer size, in bytes.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.socket-binding

The listener socket binding

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.ssl-context

Reference to the SSLContext to be used by this listener.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.ssl-session-cache-size

The maximum number of active SSL sessions

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.ssl-session-timeout

The timeout for SSL sessions, in seconds

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.tcp-backlog

Configure a server with the specified backlog.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.tcp-keep-alive

Configure a channel to send TCP keep-alive messages in an implementation-dependent manner.

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.url-charset

URL charset

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.verify-client

The desired SSL client authentication mode for SSL channels

thorntail.undertow.servers.*KEY*.https-listeners.*KEY*.worker

The listeners XNIO worker

thorntail.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}.

thorntail.undertow.servers.*KEY*.servlet-container

The servers default servlet container

thorntail.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

thorntail.undertow.servlet-containers.*KEY*.crawler-session-management-setting.session-timeout

The session timeout for sessions that are owned by crawlers

thorntail.undertow.servlet-containers.*KEY*.crawler-session-management-setting.user-agents

Regular expression that is used to match the user agent of a crawler

thorntail.undertow.servlet-containers.*KEY*.default-buffer-cache

The buffer cache to use for caching static resources

thorntail.undertow.servlet-containers.*KEY*.default-cookie-version

The default cookie version servlet applications will send

thorntail.undertow.servlet-containers.*KEY*.default-encoding

Default encoding to use for all deployed applications

thorntail.undertow.servlet-containers.*KEY*.default-session-timeout

The default session timeout (in minutes) for all applications deployed in the container.

thorntail.undertow.servlet-containers.*KEY*.directory-listing

If directory listing should be enabled for default servlets.

thorntail.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.

thorntail.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

thorntail.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.

thorntail.undertow.servlet-containers.*KEY*.eager-filter-initialization

If true undertow calls filter init() on deployment start rather than when first requested.

thorntail.undertow.servlet-containers.*KEY*.file-cache-max-file-size

The maximum size of a file that will be cached in the file cache

thorntail.undertow.servlet-containers.*KEY*.file-cache-metadata-size

The maximum number of files that will have their metadata cached

thorntail.undertow.servlet-containers.*KEY*.file-cache-time-to-live

The length of time in ms an item will stay cached. By default this is 2000 for exploded deployments, and -1 (infinite) for archive deployments

thorntail.undertow.servlet-containers.*KEY*.ignore-flush

Ignore flushes on the servlet output stream. In most cases these just hurt performance for no good reason.

thorntail.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.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.development

Enable Development mode which enables reloading JSP on-the-fly

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.disabled

Disable the JSP container.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.display-source-fragment

When a runtime error occurs, attempts to display corresponding JSP source fragment

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.dump-smap

Write SMAP data to a file.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.error-on-use-bean-invalid-class-attribute

Enable errors when using a bad class in useBean.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.generate-strings-as-char-arrays

Generate String constants as char arrays.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.java-encoding

Specify the encoding used for Java sources.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.keep-generated

Keep the generated Servlets.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.mapped-file

Map to the JSP source.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.modification-test-interval

Minimum amount of time between two tests for updates, in seconds.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.optimize-scriptlets

If JSP scriptlets should be optimised to remove string concatenation

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.recompile-on-fail

Retry failed JSP compilations on each request.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.scratch-dir

Specify a different work directory.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.smap

Enable SMAP.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.source-vm

Source VM level for compilation.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.tag-pooling

Enable tag pooling.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.target-vm

Target VM level for compilation.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.trim-spaces

Trim some spaces from the generated Servlet.

thorntail.undertow.servlet-containers.*KEY*.jsp-setting.xPowered-by

Enable advertising the JSP engine in x-powered-by.

thorntail.undertow.servlet-containers.*KEY*.max-sessions

The maximum number of sessions that can be active at one time

thorntail.undertow.servlet-containers.*KEY*.mime-mappings.*KEY*.value

The mime type for this mapping

thorntail.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

thorntail.undertow.servlet-containers.*KEY*.persistent-sessions-setting.relative-to

The directory the path is relative to

thorntail.undertow.servlet-containers.*KEY*.preserve-path-on-forward

If this is true Undertow will reset request path, URL and URI information to original values after forward.

thorntail.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.

thorntail.undertow.servlet-containers.*KEY*.session-cookie-setting.comment

Cookie comment

thorntail.undertow.servlet-containers.*KEY*.session-cookie-setting.domain

Cookie domain

thorntail.undertow.servlet-containers.*KEY*.session-cookie-setting.http-only

Is cookie http-only

thorntail.undertow.servlet-containers.*KEY*.session-cookie-setting.max-age

Max age of cookie

thorntail.undertow.servlet-containers.*KEY*.session-cookie-setting.name

Name of the cookie

thorntail.undertow.servlet-containers.*KEY*.session-cookie-setting.secure

Is cookie secure?

thorntail.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).

thorntail.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

thorntail.undertow.servlet-containers.*KEY*.use-listener-encoding

Use encoding defined on listener

thorntail.undertow.servlet-containers.*KEY*.websockets-setting.buffer-pool

The buffer pool to use for websocket deployments

thorntail.undertow.servlet-containers.*KEY*.websockets-setting.deflater-level

Configures the level of compression of the DEFLATE algorithm

thorntail.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.

thorntail.undertow.servlet-containers.*KEY*.websockets-setting.per-message-deflate

Enables websocket's per-message compression extension, RFC-7692

thorntail.undertow.servlet-containers.*KEY*.websockets-setting.worker

The worker to use for websocket deployments

thorntail.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.39. 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 about application development with OpenShift, see:

- OpenShift Interactive Learning Portal

To reduce network load and shorten the build time of your application, set up a Nexus mirror for Maven on your Minishift or CDK:

- Setting Up a Nexus Mirror for Maven

APPENDIX G. PROFICIENCY LEVELS

Each available example teaches concepts that require certain minimum knowledge. This requirement varies by example. 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 example.

Foundational

The examples 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 example.

Advanced

When using Advanced examples, the assumption is that you are familiar with the common concepts and terminology of the subject area of the example in addition to Kubernetes and OpenShift. You must also be able to perform basic tasks on your own, for example, configuring services and applications, or administering networks. If a service is needed by the example, but configuring it is not in the scope of the example, the assumption is that you have the knowledge to properly configure it, and only the resulting state of the service is described in the documentation.

Expert

Expert examples 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

Developer Launcher (developers.redhat.com/launch)

developers.redhat.com/launch called Developer Launcher is a stand-alone getting started experience provided by Red Hat. It helps you get started with cloud-native development on OpenShift. It contains functional example applications that you can download, build, and deploy on OpenShift.

Minishift or CDK

An OpenShift cluster running on your machine using Minishift.

H.2. TERMS SPECIFIC TO DEVELOPER LAUNCHER

Example

An application specification, for example *a web service with a REST API*.
Examples generally do not specify which language or platform they should run on; the description only contains the intended functionality.

Example application

A language-specific implementation of a particular example on a particular runtime. Example applications are listed in an examples catalog.
For example, an example application is a web service with a REST API implemented using the Thorntail runtime.

Examples Catalog

A Git repository that contains information about example applications.

Runtime

A platform that executes an example application. For example, Thorntail or Eclipse Vert.x.