Open Liberty 2020

Release Notes for Open Liberty 20.0.0.5 on Red Hat OpenShift Container Platform

Release Notes for Open Liberty 2020 on Red Hat OpenShift Container Platform

Last Updated: 2020-05-08

Open Liberty 2020 Release Notes for Open Liberty 20.0.0.5 on Red Hat OpenShift Container Platform

Release Notes for Open Liberty 2020 on Red Hat OpenShift Container Platform

Legal Notice

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in Open Liberty 2020 on Red Hat OpenShift Container Platform release.

Table of Contents

CHAPTER 1. FEATURES

With Open Liberty 20.0.0.5, developers can add a configurable attribute to the EJB Persistent Timer feature. The new attribute sets a maximum amount of time allowed for a persistent timer to complete before another server can take over and run the timer instead.

In Open Liberty 20.0.0.5:

- EJB persistent timers
- JAAS custom login modules
- Open Liberty logging
- Omit specified fields from JSON logging output

View the list of fixed bugs in 20.0.0.5.

1.1. RUN YOUR APPS USING 20.0.0.5

If you're using Maven, here are the coordinates:

```
<dependency>
    <groupId>io.openliberty</groupId>
    <artifactId>openliberty-runtime</artifactId>
    <version>20.0.0.5</version>
    <type>zip</type>
</dependency>
```

Or for Gradle:

```
dependencies {
    libertyRuntime group: 'io.openliberty', name: 'openliberty-runtime', version: '[20.0.0.5,)'
}
```

Or if you're using Docker:

```
FROM open-liberty
```

1.1.1. EJB persistent timers coordination and failover across members

Prior to this feature, coordination of automatic EJB persistent timers across multiple Open Liberty servers was limited to ensuring that only a single instance of a timer is created across all servers by configuring the EJB timer service on each to persist timers to the same database. This caused a single timer instance to be created on one of the servers but without the ability to go to another server if the original server stops or crashes. To enable failover, this feature adds a new configurable attribute, **missedTaskThreshold**, which specifies the maximum amount of time that you want to allow for an execution of a persistent timer to complete before allowing another server to take over and run it instead.

Enable the EJB persistent timers feature or another feature that implicitly enables it, such as **ejb-3.2**, and configure it to use a data source. In this example, we configure the feature to use the Java EE or Jakarta EE default data source. This much configuration is required regardless of whether you want to enable failover.

Add the feature to the **server.xml**:

```
<server>
  <featureManager>
    <feature>ejbPersistentTimer-3.2</feature>
    <feature>jdbc-4.2</feature>
    ... other features
  </featureManager>

  <dataSource id="DefaultDataSource">
    <jdbcDriver libraryRef="OraLib"/>
    <properties.oracle URL="jdbc:oracle:thin:@//localhost:1521/EXAMPLEDB"/>
    <containerAuthData user="dbuser" password="dbpwd"/>
  </dataSource>
  <library id="OraLib">
    <file name="${shared.resource.dir}/jdbc/ojdbc8.jar" />
  </library>

  <!-- The following enables failover for persistent timers -->
  <persistentExecutor id="defaultEJBPersistentTimerExecutor" missedTaskThreshold="5m"/>

  ...
</server>
```

1.2. LOAD JAAS LOGINMODULES FROM RESOURCE ADAPTERS

Open Liberty supports JAAS LoginModules for configuring access to JCA-managed resources. In general, JAAS LoginModules are packaged as a shared library and configured for Open Liberty. Sometimes a JAAS LoginModule is packaged as part of a JCA ResourceAdapter. In the past, to use these JAAS LoginModules, the classes from the ResourceAdapter had to be extracted and configured as a shared library. With this new feature, the **jaasLoginModule** can now load the classes directly from the resource adapter, simplifying the configuration. Before, it was necessary to package (or repackage) JAAS custom login modules into a separate shared library to configure Open Liberty to use them.

Enable the Application Security and JCA features and configure a resource adapter. Use the **classProviderRef** attribute on the **jaasLoginModule** element to reference the **id** of the resource adapter:

```
<server>
  <featureManager>
    <feature>appSecurity-2.0</feature>
    <feature>jca-1.7</feature>
    ... other features
  </featureManager>

  <resourceAdapter id="eciResourceAdapter" location="${shared.resource.dir}/cicseci.rar"/>

  <!-- classProviderRef indicates that the login module class is found in the resource adapter -->
  <jaasLoginModule id="identityProp" controlFlag="REQUIRED"
     className="com.ibm.ctg.security.idprop.LoginModule"
     classProviderRef="eciResourceAdapter">
    <options propIdentity="Caller"/>
  </jaasLoginModule>

  <jaasLoginContextEntry id="CTGEntry" loginModuleRef="identityProp" name="CTGEntry"/>
```

```
  <connectionFactory id="cf1" jndiName="eis/cf1" jaasLoginContextEntryRef="CTGEntry">
    <properties.eciResourceAdapter ConnectionUrl="tcp://localhost" portNumber="2006"
serverName="MYSERVER"/>
  </connectionFactory>

  ...
</server>
```

The same approach can be used for JAAS custom login modules that are packaged within an application. Set the **classProviderRef** to point to the **id** of the **application**, **webApplication**, or **enterpriseApplication** element that contains the login module class. When packaging JAAS custom login modules within an application, include the login module within one of the following places:

- Within a top level JAR of the enterprise application.
- Within a resource adapter module of the enterprise application.
- Within the web module of the enterprise application.
- Within an EJB module of the enterprise application.
- Within a web application.

It should be noted that JAAS custom login modules require the use of a resource reference with container-managed authentication.

You can find out more about Configuring a JAAS custom login module for Liberty .

1.2.1. Open Liberty console logging now has the ability to format logs with date and time stamps and other relevant information

In Open Liberty, users can apply different formats, such as JSON or dev, to the server logs that appear in their **console.log** file by using the **consoleFormat** logging attribute in the server logging configuration. The dev format is the default format and shows messages in a basic format, with no timestamp or any other relevant information. It only shows the message log level and the message itself.

For example:

```
consoleFormat=dev (default)
[AUDIT ] CWWKE0001I: The server server1 has been launched.
```

This feature introduces a new option called **simple** for the **consoleFormat** logging server configuration attribute. This new option configures Open Liberty to output logs in the same simple format used in the **message.log** file, with date/time stamps and other relevant information, to the **console.log** file or to the console (**console.log**/**standard-out**).

For example:

```
consoleFormat=simple
[25/11/19 10:02:30:080 EST] 00000001 com.ibm.ws.kernel.launch.internal.FrameworkManager A
CWWKE0001I: The server server1 has been launched.
```

To configure the Open Liberty logs to output logs in the new simple console format, you just have to set the following logging server configuration in **server.env**, **bootstrap.properties**, or **server.xml**:

1.2.1.1. server.env

WLP_LOGGING_CONSOLE_FORMAT=simple

1.2.1.2. bootstrap.properties

com.ibm.ws.logging.console.format=simple

1.2.1.3. server.xml

<logging consoleFormat="simple"/>

1.2.2. Omit specified fields from JSON logging output

In Open Liberty, users can format their server logs in JSON format. When logs are in JSON format, users have to specify the sources (message, trace, accessLog, ffdc, audit) they want to send to **messages.log** or **console.log/standard-out**.

Users can now specify the JSON fields they want to omit. This feature adds an option for users to omit JSON fields in the JSON logging process. The option to omit JSON field names in Open Liberty is extremely useful, as users might not want certain default fields provided by Open Liberty in their JSON output. Undesired fields add to the size of the records, which wastes network I/O during record transmissions and wastes space in downstream log aggregation tools. Now, users can choose to emit only the fields they need so they can send to downstream log aggregation tools without using more space and I/O than necessary. For example, someone who's running Open Liberty in Docker containers, with a single server in each container, might not want to include the JSON fields that represent the server name and user directory.

The attribute was initially used only for renaming field names. To rename a JSON field name, the format is specified as **source:defaultFieldName:newFieldName** or **defaultFieldName:newFieldName**. To omit **defaultFieldName**, leave **newFieldName** empty. For example, to omit a field for all sources, use the **defaultFieldName:** format. To omit a field for a specific source, use the **source:defaultFieldName:** format, where **source** is the source you want to specify, such as message, trace, accessLog, ffdc, or audit.

Adding the following example to **bootstrap.properties** omits JSON fields:

```
com.ibm.ws.logging.json.field.mappings=trace:ibm_userDir: ,ibm_datetime:
```

You can find more information by going to Logging and Trace on IBM Knowledge Center or by visiting the Open Liberty logging documentation .

CHAPTER 2. RESOLVED ISSUES

See the Open Liberty 20.0.0.5 issues that were resolved for this release .

CHAPTER 3. FIXED CVES

For a list of CVEs that were fixed in Open Liberty 20.0.0.5, see security vulnerabilities.

CHAPTER 4. KNOWN ISSUES

See the list of issues that were found but not fixed during the development of 20.0.0.5 .