Open Liberty 2020

Release Notes for Open Liberty 20.0.0.3 on Red Hat OpenShift Container Platform

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

Last Updated: 2020-04-06

Open Liberty 2020 Release Notes for Open Liberty 20.0.0.3 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

In Open Liberty 20.0.0.3, you can now access Kafka specific properties such as the message key and message headers, not just the payload of the message, as was the case previously with the MicroProfile Reactive Messaging Message API. Also, you can now set the SameSite attribute on the Session cookie, the LTPA, and JWT cookies as well as application-defined cookies.

In Open Liberty 20.0.0.3:

- Kafka specific properties
- Adding the SameSite cookie attribute

View the list of fixed bugs in 20.0.0.3.

1.1. RUN YOUR APPS USING 20.0.0.3

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

```
<dependency>
    <groupId>io.openliberty</groupId>
    <artifactId>openliberty-runtime</artifactId>
    <version>20.0.0.3</version>
    <type>zip</type>
</dependency>
```

Or for Gradle:

```
dependencies {
    libertyRuntime group: 'io.openliberty', name: 'openliberty-runtime', version: '[20.0.0.3,)'
}
```

Or if you're using Docker:

```
FROM open-liberty
```

1.2. KAFKA SPECIFIC PROPERTIES

New to Open Liberty is Kafka specific properties. The basic MicroProfile Reactive Messaging **Message** API does not let the user access anything other than the payload of the message. The native Kafka client API allows the user to access some Kafka specific message properties, such as the message key and message headers.

1.2.1. Incoming Messages

For incoming messages, we have now allowed the user to unwrap a Message to gain access to the underlying **ConsumerRecord**.

```
@Incoming("channel1")
public CompletionStage<Void> consume(Message<String> message) {
    ConsumerRecord<String, String> cr = (ConsumerRecord<String, String>)
message.unwrap(ConsumerRecord.class);
    String key = consumerRecord.key();
```

```
    String value = consumerRecord.value();
    String topic = consumerRecord.topic();
    int partition = consumerRecord.partition();
    long timestamp = consumerRecord.timestamp();
    Headers headers = consumerRecord.headers();
    // some more code....
    return CompletableFuture.completedFuture(null);
}
```

1.2.2. Outgoing Messages

For outgoing messages, if the payload is a ProducerRecord then the properties within it are passed on to Kafka.

```
@Outgoing("channel2")
public Message<ProducerRecord> publish() throws UnsupportedEncodingException {
  ProducerRecord<String, String> producerRecord = new ProducerRecord<String, String>
("myTopic", null, "myKey", "myValue");
  producerRecord.headers().add("HeaderKey", "HeaderValue".getBytes("UTF-8"));
  return Message.of(producerRecord);
}
```

The example above assumes that no topic has been explicitly pre-configured in the MicroProfile Config for the channel. If the topic is pre-configured then that will take precedence and the topic in the ProducerRecord will be ignored.

1.2.3. Example using a pre-configured topic

In this example, the topic is pre-configured using MicroProfile Config to be **myOtherTopic** so the topic set in the ProducerRecord is ignored.

MicroProfile Config Properties

```
mp.messaging.outgoing.channel3.connector=liberty-kafka
mp.messaging.outgoing.channel3.topic=myOtherTopic #Overrides value in code
```

Reactive Messaging Bean

```
@Outgoing("channel3")
public Message<ProducerRecord<K, V>> publish() {
  ProducerRecord pr = new ProducerRecord("myTopic", "myValue");
  return Message.of(pr);
}
```

1.3. ADDING THE SAMESITE COOKIE ATTRIBUTE

Open Liberty now offers the ability to set the SameSite attribute on the Session cookie, the LTPA, and JWT cookies as well as application-defined cookies. The SameSite attribute can be added by adding one or more **server.xml** configuration options.

The Servlet **javax.servlet.http.Cookie** API does not offer the ability to set the SameSite attribute on a Cookie. If the SameSite attribute is needed, the options for setting it are currently limited to using the **HttpServletResponse.addHeader** and **HttpServletResponse.setHeader** and constructing the Set-

Cookie header. The SameSite attribute is used by browsers to determine if a particular cookie should be sent with a request.

There are three values for the SameSite attribute: **Lax**, **Strict**, **None**.

- **SameSite=Strict**: The cookie is only sent with "same-site" requests. The cookie is only sent by the web browser if the site for the cookie matches the site in the address bar for example.
- **SameSite=Lax**: The cookie is sent with both "same-site" and "cross-site" top-level navigation requests. Clicking a link for example.
- **SameSite=None**: The cookie is sent with both "same-site" and "cross-site" requests. The cookie is sent by the web browser for third party contexts, embedded content for example.

To use SameSite cookie attribute:

1. Set the Session Cookie SameSite attribute using the following **server.xml** configuration: **<httpSession cookieSameSite="Disabled|Strict|Lax|None"/>**
2. The default value is **Disabled**. This means no SameSite attribute will be added. Set the **LTPA/JWT** Cookie SameSite attribute using the following server.xml configuration: **<webAppSecurity sameSiteCookie="Disabled|Strict|Lax|None"/>**
3. The default value is **Disabled**. This means no SameSite attribute will be added. Set the SameSite attribute on Cookies via the following server.xml configuration:

```
<httpEndpoint id="defaultHttpEndpoint"
              httpPort="9080"
              httpsPort="9443" >
   <samesite lax="cookieOne" strict="cookieTwo" none="cookieThree"/>
</httpEndpoint>
```

The **<httpEndpoint/>** SameSite configuration allows the use of wildcards in the following ways:

A standalone wildcard (*). All cookies would have the SameSite=Lax attribute. This includes the Session and LTPA/JWT cookies unless the **<httpSession/>** and/or **<webAppSecurity/>** configuration has also been set.

```
<httpEndpoint id="defaultHttpEndpoint"
              httpPort="9080"
              httpsPort="9443" >
   <samesite lax="*" />
</httpEndpoint>
```

At the end of one or more cookie names. The below snippet would map the following cookie name to SameSite attributes:

- cookieOne → SameSite=Lax
- cookieTwo → SameSite=Strict
- cookieThree → SameSite=None

```
<httpEndpoint id="defaultHttpEndpoint"
              httpPort="9080"
              httpsPort="9443" >
```

```
    <samesite lax="cookie*" strict="cookieTwo" none="cookieThree"/>
</httpEndpoint>
```

The **<httpSession/>** and **<webAppSecurity/>** configuration takes precedence over the **<httpEndpont/>configuration**.

When a cookie matches the **SameSite=None** configuration then the **Secure** attribute will be automatically added to the cookie.

The **<httpEndpoint/>** configuration can apply to any **Set-Cookie** header.

Technical details regarding the SameSite attribute can be found in the following RFC: Cookies: HTTP State Management Mechanism

CHAPTER 2. RESOLVED ISSUES

See the Open Liberty 20.0.0.3 issues that were resolved for this release .

CHAPTER 3. FIXED CVES

For a list of CVEs that were fixed in Open Liberty 20.0.0.3, see security vulnerabilities.

CHAPTER 4. KNOWN ISSUES

See the list of issues that were found but not fixed during the development of 20.0.0.3 .