Open Liberty 2021

Release Notes for Open Liberty 21.0.0.2 on Red Hat OpenShift Container Platform

Release Notes for Open Liberty 2021 on Red Hat OpenShift Container Platform

Last Updated: 2021-02-23

Open Liberty 2021 Release Notes for Open Liberty 21.0.0.2 on Red Hat OpenShift Container Platform

Release Notes for Open Liberty 2021 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 2021 on Red Hat OpenShift Container Platform release.

Table of Contents

CHAPTER 1. FEATURES

With Open Liberty 21.0.0.2 you can now configure **trustedHeaderOrigin** and **trustedSensitiveHeaderOrigin** using IP address wildcards and hostnames.

In Open Liberty 21.0.0.2:

- trustedHeaderOrigin Configuration Improvements
- Significant bugs fixed in this release

Run Open Liberty 21.0.0.2 now

View the list of fixed bugs in 21.0.0.2. '''

1.1. HTTP CHANNEL CONFIGURATION IMPROVEMENTS

"WebSphere private headers" in the form **$WSXX** are used by WebSphere-aware proxies to provide information about original requests. The values provided by these headers are made available to the application via **ServletRequest** APIs such as **getRemoteHost()**. Two HTTP Dispatcher custom properties are provided by Open Liberty to control which remote hosts are allowed to send the **trustedHeaderOrigin** and **trustedSensitiveHeaderOrigin** private headers. Ideally, these dispatcher properties should be configured to only trust known proxy servers which will be forwarding requests to the Open Liberty server.

Previously, **trustedHeaderOrigin** and **trustedSensitiveHeaderOrigin** only accepted " * " , "none", or a list of full IP addresses (eg. " 127.0.0.1, 192.168.6.6 "). As requested by customers, both properties can now additionally be configured with IP address wildcards and hostnames. As an example, the following list which will be valid for either property: "localhost, 127.0.0.1, 192.168. * . * , 0:0:0:0:0:ffff*:*, *.ibm.com "

The same list as a server.xml example:

```
<httpDispatcher trustedHeaderOrigin="*" trustedSensitiveHeaderOrigin="localhost, 127.0.0.1,
192.168.*.*, 0:0:0:0:0:ffff:*:*, *.ibm.com"/>
```

For more information about these properties, see Open Liberty docs on httpDispatcher

1.2. SIGNIFICANT BUGS FIXED IN THIS RELEASE

We’ve spent some time fixing bugs. The following sections describe just some of the issues resolved in this release. If you’re interested, here’s the full list of bugs fixed in 21.0.0.2 .

- Update gRPC dependencies to 1.35
  Currently Open Liberty uses **grpc-java** version 1.31.1 to provide its **grpc-1.0** and **grpcClient-1.0** features. This issue will update the **grpc-java** dependencies we build on to their latest version which is 1.35.0. The 1.35.0 release we’re pulling in provides numerous improvements and bug fixes; for a full list see GRPC Java Release Notes

- Expression Language 3.0 value lookup performance improvement
  Lookup performance has been improved for certain Expression Language 3.0 values. By optimizing EL 3.0 **context handling** and **classloader** calls, the evaluations of EL unscoped and static expressions perform better with this fix. This fix includes Apache bug fix 62453 and also Apache bug 63781 to support multiple java levels correctly.

- System Web Application Bundles may not be fully provisioned until after the server reports as started
  When a Web Application Bundle (WAB) is included in a Liberty feature (also called a system WAB) it is possible that the server started message and the TCP ports will be opened and listening before the WAB has been fully provisioned with the web container. This is different behaviour to web applications WARs where the server started and TCP ports are delayed until all the applications have started or a timeout has occurred on server start.

  This could lead to 404 errors early after the server started status has returned until the WABs come online with the web container. The server should delay the TCP ports listening and the server started event for system WABs similar to the way it waits for applications to start.

  This has been fixed by delaying the server started message and TCP port listening until the system WABs are ready to serve, or a timeout has occurred. This has been done by introducing a new service type **ServerReadyStatus**, the feature manager will call this service before registering **ServerStarted**. Implementations of **ServerReadyStatus** have the check method called to allow them to block while waiting for the server to be ready before proceeding to register **ServerStarted**.

  This allows the **WABInstaller** to check on the status of the system WABs to make sure the have completed their provisioning to the web container.

- FeatureUtility not parsing liberty custom environment variables
  When the entry is /**${shared.config.dir}/server.xml”** and the environment variable is **WLP_USER_DIR=/test/** in the **test/config/server.xml** file the user recieves an error stating that "it couldn’t find file at /user/test/wlp/usr/shared/config/server.xml" where it should have been looking for **test/config/server.xml**.

  This bug caused include tags to not be parsed correctly by not replacing the environment variables found here

- NullPointerException in HttpServletRequest or HttpServletResponse context proxies
  Using the **HttpServletRequest** or **HttpServletResponse** JAX-RS context proxies with a null runtime context previously may have resulted in a **NullPointerException** similar to the following:

  ```
  java.lang.NullPointerException
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:90)
  at
  sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:55)
  at java.lang.reflect.Method.invoke(Method.java:508)
  at
  com.ibm.ws.jaxrs20.injection.HttpServletRequestInjectionProxy$1.invoke(HttpServletRequestI
  njectionProxy.java:58)
  ```

  This issue was resolved by adding a null checks to these two proxy classes similar to those in other proxy classes.

- ClassCastException when serving a static resource
  When the following conditions occur:

- The **servlet-4.0** feature is installed

- The request is for a static resource

- The application wraps the request object in its filter.
  A **java.lang.ClassCastException** might occur near the end of the request cycle when the server is trying to cache information for a served static resource. (e.g. xml, html, jpg, css...). The response is still sent back to a client normally without any issue.

  An example of the exception:

  ```
  java.lang.ClassCastException: com.example.MyWrappedRequest incompatible with
  com.ibm.ws.webcontainer40.srt.SRTServletRequest40
  at com.ibm.ws.webcontainer40.servlet.CacheServletWrapper40.
  (CacheServletWrapper40.java:57)
  at
  com.ibm.ws.webcontainer40.servlet.factory.CacheServletWrapperFactory40Impl.createCacheS
  ervletWrapper(CacheServletWrapperFactory40Impl.java:30)
  at com.ibm.ws.webcontainer.WebContainer.addToCache(WebContainer.java:1231)
  at
  com.ibm.ws.webcontainer.extension.DefaultExtensionProcessor.handleRequest(DefaultExtensi
  onProcessor.java:538)
  at
  com.ibm.ws.webcontainer.filter.WebAppFilterChain.invokeTarget(WebAppFilterChain.java:18
  2)
  ```

  This bug has been removed by fixing a **ClassCastException** when caching a static servlet wrapper under **servlet-4.0** feature.

1.3. RUN YOUR APPS USING 21.0.0.2

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

```
<dependency>
    <groupId>io.openliberty</groupId>
    <artifactId>openliberty-runtime</artifactId>
    <version>21.0.0.2</version>
    <type>zip</type>
</dependency>
```

Or for Gradle:

```
dependencies {
    libertyRuntime group: 'io.openliberty', name: 'openliberty-runtime', version: '[21.0.0.2,)'
}
```

Or if you're using Docker:

```
FROM open-liberty
```

CHAPTER 2. RESOLVED ISSUES

See the Open Liberty 21.0.0.2 issues that were resolved for this release .

CHAPTER 3. FIXED CVES

For a list of CVEs that were fixed in Open Liberty 21.0.0.2, see security vulnerabilities.

CHAPTER 4. KNOWN ISSUES

See the list of issues that were found but not fixed during the development of 21.0.0.2 .