Red Hat build of Eclipse Vert.x 3.9

Release Notes for Eclipse Vert.x 3.9

For use with Eclipse Vert.x 3.9.6

Last Updated: 2021-03-09

Red Hat build of Eclipse Vert.x 3.9 Release Notes for Eclipse Vert.x 3.9

For use with Eclipse Vert.x 3.9.6

Legal Notice

Abstract

This Release Note contains important information related to Eclipse Vert.x 3.9.6

Table of Contents

PREFACE

Date of release: 2021-03-11

RED HAT BUILD OF ECLIPSE VERT.X 3.9 - END OF LIFE

The Red Hat build of Eclipse Vert.x 3.9 is the last supported release of the major version 3.x. The full support life cycle of Eclipse Vert.x 3.x version ends on March 31, 2021. See the product life cycle page for details.

It is recommended that you migrate your applications to Red Hat build of Eclipse Vert.x 4.

See the Eclipse Vert.x 4 documentation for more information.

The *Eclipse Vert.x 4.0 Migration Guide* provides information on how to upgrade your Eclipse Vert.x 3.x applications to Eclipse Vert.x 4.

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.

CHAPTER 1. REQUIRED INFRASTRUCTURE COMPONENT VERSIONS

Red Hat does not provide support for components listed below, with the exception of components explicitly designated as supported.

Component name	Version
Maven	3.6.0
Fabric8 Maven Plugin	4.4.1
JDK[a][b]	OpenJDK 8, OpenJDK 11[c]
Red Hat Enterprise Linux 7[d]	7.7
Red Hat Enterprise Linux 8[e]	8.1
OpenShift Container Platform (OCP)[f]	3.11, 4.6, 4.7
Minishift	1.34.2 or later
CDK[g]	3.11.0
git	2.0 or later
oc command line tool	3.11 or later[h]

[a] A full JDK installation is required, as JRE does not provide tools for compiling Java applications from source.

[b] Red Hat OpenJDK is supported by Red Hat.

[c] Red Hat supports only LTS releases of JDK.

[d] For deploying applications based on CNR on stand-alone RHEL in a production environment.

[e] For deploying applications based on CNR on stand-alone RHEL in a production environment.

[f] OCP is supported by Red Hat

[g] CDK is supported by Red Hat

[h] The version of the **oc** CLI tool should correspond to the version of OCP that you are using.

CHAPTER 2. SUPPORTED ECLIPSE VERT.X RUNTIME COMPONENT CONFIGURATIONS AND INTEGRATIONS

The following resource defines the supported configurations and integrations of Red Hat products with Eclipse Vert.x:

- For a list of technologies that are supported for integration with Eclipse Vert.x in production environments see the Supported Eclipse Vert.x configurations and integrations.
- For a list of Eclipse Vert.x runtime artifacts and their versions see the component details page.

CHAPTER 3. FEATURES

3.1. NEW AND CHANGED FEATURES

This section describes the new functionalities introduced in this release. It also contains information about changes in the existing functionalities.

3.1.1. Changes in EventBus JavaScript client

The following section explains the changes in EventBus JavaScript client.

3.1.1.1. EventBus JavaScript client available only in npm

In releases before Eclipse Vert.x 3.9.5, the event bus JavaScript client was available in various locations, such as Maven Central, NPM, and so on.

From Eclipse Vert.x 3.9.5, the EventBus JavaScript client module is available in a new npm location. The client is not available in other locations. You should update your build systems to use the module from the following new location.

3.1.1.2. Versioning of JavaScript client

In releases before Eclipse Vert.x 3.9.5, every Eclipse Vert.x release included a new release of the JavaScript client.

However, from Eclipse Vert.x 3.9.5 onward, a new version of JavaScript client will be available in npm only if there changes in the client. You do not need to update your client application for every Eclipse Vert.x release, unless there is a version change.

3.1.2. Value accessor methods in `Row` and `Tuple` thrown exceptions for invalid values

The value accessor methods in **Row** and **Tuple** throw exceptions instead of returning **null** for invalid values.

- The methods in **Row** throws the **NoSuchElementException** when a lookup for a given column does not exist.
- The methods in **Row** and **Tuple** throw the **ClassCastException** when the returned value cannot be casted or coerced to the expected type.

3.1.3. Wildcard port (0) shared with several instances of the same Verticle

When several instances of the same verticle use the wildcard port (0), the first instance binds to a random port and all the other instances also bind to the same random port.

In releases prior to Eclipse Vert.x 3.9.2, all the instances of the same verticle that used the wildcard port (0) were bound to different random ports.

3.1.4. Prepared statement created using the `SqlConnection.Prepare` method is not cached

The prepared statements that are created using the **SqlConnection.Prepare** method are no longer cached internally. If you want to cache the prepared statements, you must set up the caching.

3.1.5. PgConnection.Prepare method creates named prepared statement

The **PgConnection.Prepare** method creates a named prepared statement. In the previous releases, the method created unnamed prepared statements.

3.1.6. Support for Eclipse Vert.x Runtime on IBM Z and IBM Power Systems

The Red Hat build of Eclipse Vert.x for s390x and ppc64le platforms is supported only in OpenShift environments provisioned on IBM Z and IBM Power Systems infrastructure. Running an Eclipse Vert.x application on a stand-alone installation of RHEL on IBM Z and IBM Power Systems is not supported.

Eclipse OpenJ9 Java images for IBM Z, IBM Power Systems and new images for products supported on IBM Z and IBM Power Systems are available in the Red Hat Ecosystem Catalog .

3.1.7. Deploying example applications on OpenShift provisioned on IBM Z and IBM Power Systems infrastructure

To deploy the example applications on OpenShift environments provisioned on IBM Z and IBM Power Systems infrastructure, specify the relevant IBM Z or IBM Power Systems image name in the **pom.xml** file and commands.

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 or IBM Power Systems image names in the YAML file of the example applications.

3.1.8. Fluent Query Method

The new class Query is a fluent API. It enables you to create and configure queries before their execution. All the query collectors are available in the **Query** interface.

The new class **PreparedStatement** enables you to create and execute prepared statements. The prepared statements can execute multiple interactions such as cursor or stream operations.

The existing class **PreparedQuery** extends the new class **Query**. The class can now be used outside the context of the connection.

For example, use the following sample codes to create queries:

- Collector Query

```
PreparedQuery<RowSet<Row>> query = client.preparedQuery(sql);
PreparedQuery<SqlResult<List<Row>> collectedQuery = query.collecting(Collectors.toList());
collectedQuery.execute(tuple, ar -> ...);

// Or fluently
client.preparedQuery(sql).collecting(Collectors.toList()).execute(tuple, ar -> ...);
```

- Prepared query

```
connection.prepare(sql, ar1 -> {
  if (ar1.succeded()) {
    PreparedStatement ps = ar1.result();
    PreparedQuery<RowSet<Row>> pq = ps.query();
    pq.execute(tuple, ar2 -> ...);
```

```
    // Or fluently
      ps.query().execute(tuple, ar2 -> ...);
    }
  });
```

3.2. DEPRECATED FEATURES

The following functionalities have been deprecated in this release.

3.2.1. Deprecating the keep alive time seconds methods in Mqtt client options

The **MqttClientOptions.getKeepAliveTimeSeconds()** and **MqttClientOptions.setKeepAliveTimeSeconds()** methods have been deprecated. Use the **MqttClientOptions.getKeepAliveInterval()** and **MqttClientOptions.setKeepAliveInterval()** methods instead to get and set the keep alive interval in seconds.

3.2.2. Deprecated the net socket method in HTTP server request

The **HttpServerRequest.netSocket()** method has been deprecated. Use the new method **HttpServerRequest.toNetSocket()** instead to establish a TCP tunnel with the client.

The following example shows you how the deprecated method **HttpServerRequest.netSocket()** was used.

```
NetSocket sock = request.netSocket();
```

The following example shows you how to use the new method **HttpServerRequest.toNetSocket()**.

```
request.response().toNetSocket(ar -> {
  if (ar.succeeded()) {
    NetSocket sock = ar.result();
  }
```

3.2.3. Deprecated the upgrade method in HTTP server request

The **HttpServerRequest.upgrade()** method has been deprecated. Use the new method **HttpServerResponse.toWebSocket()** instead to upgrade the connection of the current request to a WebSocket.

The following example shows you how the deprecated method **HttpServerRequest.upgrade()** was used.

```
ServerWebSocket ws = request.upgrade();
```

The following example shows you how to use the new method **HttpServerResponse.toWebSocket()**.

```
request.response().toWebSocket(ar -> {
  if (ar.succeeded()) {
    ServerWebSocket ws = ar.result();
  }
});
```

3.2.4. Deprecated the sockjs.BridgeOptions class

The **sockjs.BridgeOptions** class has been deprecated. Use the new **sockjs.SockJSBridgeOptions** class instead. The **sockjs.SockJSBridgeOptions** class contains all the options that are required to configure the event bus bridge.

There is no change in the behavior of the new class, except that the name of the data object class has changed. When new bridges were added, there were a lot of duplicate configurations. The new class contains all the possible common configurations, and removes duplicate configurations.

3.2.5. Verticle start and stop methods with Future parameter have been deprecated

The verticle methods **start(Future<Void>)** and **stop(Future<Void>)** have been deprecated. Use the methods **start(Promise<Void>)** and **stop(Promise<Void>)** to work with promise.

3.2.6. Extension mechanism for DNS is disabled by default

The extension mechanism for DNS is disabled by default in Eclipse Vert.x. To enable EDNS use the method **AddressResolverOptions.setOptResourceEnabled()**. Specify the input parameter **optResourceEnabled** as **true**.

3.2.7. New connection handler methods

The following new handler methods are available:

- The **HttpClientRequest.connectionHandler()** method is deprecated and will be removed in a future release. Use **HttpClient.connectionHandler()** method instead to call connection handlers for client requests in your application. For example, use the following code to work with **HttpClient.connectionHandler()** method:

```
client.connectionHandler(conn -> {
 // Connection related code
});
```

- **Future<T>.setHandler()** method is deprecated and will be removed in a future release. Use **Future<T>.onComplete()**, **Future<T>.onSuccess()**, and **Future<T>.onFailure()** methods instead to call handlers on completion, success, and failure results of an action. For example, use the following code to work with **Future<T>.onComplete()** method:

```
Future<String> fut = getSomeFuture();
fut.onComplete(ar -> ...);
```

3.2.8. AdminUtils Class is no longer available

The **AdminUtils** class is no longer available. Use the new **KafkaAdminClient** class instead to perform administrative operations on a Kafka cluster.

3.2.9. Updates in HTTP Methods for WebSocket

The updates in the HTTP methods for WebSocket are:

- The usage of the term *WebSocket* in method names was inconsistent. The method names had incorrect capitalization, for example, *Websocket*, instead of *WebSocket*. The methods that had inconsistent usage of *web socket* in the following classes have been deprecated and will be

removed in a future release. Use the new methods that have correct capitalization instead.

- The following methods in **HttpServerOptions** class are deprecated.

Deprecated Methods	New Methods
getMaxWebsocketFrameSize()	**getMaxWebSocketFrameSize()**
setMaxWebsocketFrameSize()	**setMaxWebSocketFrameSize()**
getMaxWebsocketMessageSize()	**getMaxWebSocketMessageSize()**
setMaxWebsocketMessageSize()	**setMaxWebSocketMessageSize()**
getPerFrameWebsocketCompressionSupported()	**getPerFrameWebSocketCompressionSupported()**
setPerFrameWebsocketCompressionSupported()	**setPerFrameWebSocketCompressionSupported()**
getPerMessageWebsocketCompressionSupported()	**getPerMessageWebSocketCompressionSupported()**
setPerMessageWebsocketCompressionSupported()	**setPerMessageWebSocketCompressionSupported()**
getWebsocketAllowServerNoContext()	**getWebSocketAllowServerNoContext()**
setWebsocketAllowServerNoContext()	**setWebSocketAllowServerNoContext()**
getWebsocketCompressionLevel()	**getWebSocketCompressionLevel()**
setWebsocketCompressionLevel()	**setWebSocketCompressionLevel()**
getWebsocketPreferredClientNoContext()	**getWebSocketPreferredClientNoContext()**
setWebsocketPreferredClientNoContext()	**setWebSocketPreferredClientNoContext()**
getWebsocketSubProtocols()	**getWebSocketSubProtocols()**
setWebsocketSubProtocols()	**setWebSocketSubProtocols()**

  The new methods for WebSocket subprotocols use **List<String>** data type instead of a comma separated string to store items.

- The following methods in **HttpClientOptions** class are deprecated.

Deprecated Methods	New Methods
getTryUsePerMessageWebsocketCompression()	**getTryUsePerMessageWebSocketCompression()**
setTryUsePerMessageWebsocketCompression()	**setTryUsePerMessageWebSocketCompression()**
getTryWebsocketDeflateFrameCompression()	**getTryWebSocketDeflateFrameCompression()**
getWebsocketCompressionAllowClientNoContext()	**getWebSocketCompressionAllowClientNoContext()**
setWebsocketCompressionAllowClientNoContext()	**setWebSocketCompressionAllowClientNoContext()**
getWebsocketCompressionLevel()	**getWebSocketCompressionLevel()**
setWebsocketCompressionLevel()	**setWebSocketCompressionLevel()**
getWebsocketCompressionRequestServerNoContext()	**getWebSocketCompressionRequestServerNoContext()**
setWebsocketCompressionRequestServerNoContext()	**setWebSocketCompressionRequestServerNoContext()**

  - The following handler methods in **HttpServer** class are deprecated.

Deprecated Methods	New Methods
websocketHandler()	**webSocketHandler()**
websocketStream()	**webSocketStream()**

- **WebsocketRejectedException** is deprecated. The methods will throw the **UpgradeRejectedException** instead.

3.2.10. Deprecated authentication and authorization classes and methods

The following authentication and authorization classes and methods are deprecated and will be replaced in a future release.

- Classes:
  - **AbstractUser**
  - **PubSecKeyOptions**
  - **JDBCAuthOptions**

- **JDBCHashStrategy**
- **AccessToken**
- **KeycloakHelper**
- **ShiroAuth**
- **AuthProviderInternal**

- Methods:
  - **User.isAuthorized()**
  - **User.clearCache()**
  - **User.setAuthProvider()**
  - **Oauth2Auth.introspectToken()**
  - **Oauth2Auth.getFlowType()**
  - **Oauth2Auth.loadJWK()**
  - **Oauth2Auth.rbacHandler()**
  - **Oauth2ClientOptions.isUseBasicAuthorization()**
  - **Oauth2ClientOptions.setUseBasicAuthorizationHeader()**
  - **Oauth2ClientOptions.getScopeSeparator()**
  - **Oauth2ClientOptions.setScopeSeparator()**

3.2.11. Methods to create clients that have no shared data sources

Use the following new methods to create clients that do not have shared data sources with other clients. These methods maintain their own data sources.

Deprecated Methods	New Methods
MongoClient.createNonShared()	**MongoClient.create()**
JDBCClient.createNonShared()	**JDBCClient.create()**
CassandraClient.createNonShared()	**CassandraClient.create()**
MailClient.createNonShared()	**MailClient.create()**

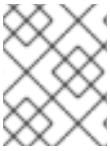
CHAPTER 4. RELEASE COMPONENTS

4.1. SUPPORTED ARTIFACTS INTRODUCED IN THIS RELEASE

No artifacts have been moved form Technology Preview to fully supported in this release:

4.2. TECHNOLOGY PREVIEW ARTIFACTS INTRODUCED IN THIS RELEASE

No new artifacts are provided as Technology Preview in this release.



NOTE

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

4.3. ARTIFACTS REMOVED IN THIS RELEASE

No artifacts are removed in this release.

4.4. ARTIFACTS DEPRECATED IN THIS RELEASE

No artifacts are marked as deprecated in this release.

CHAPTER 5. FIXED ISSUES

This Eclipse Vert.x release incorporates all bugfixes from community release of versions 3.9.6. Issues resolved in the community releases are listed in the Eclipse Vert.x 3.9.6 community release notes .

5.1. RXJAVA1 AND RXJAVA2 DEPENDENCIES ARE ADDED BY DEFAULT IN THE POM FILE

In Eclipse Vert.x 3.9.5, **vertx-rx-java** and **vertx-rx-java2** modules were marked as optional dependencies in the POM files. However, by default, the Eclipse Vert.x JUnit5 extension code loaded the Rx versions of **Vertx** and **WebClient** classes. Since these dependencies were not available, exceptions were thrown.

This issue has been fixed in Eclipse Vert.x 3.9.6. The RxJava1 and RxJava2 dependencies have been added by default in the POM file.

5.2. THE STARR.VERSION PROPERTY IS NO LONGER INDICATED AS MISSING IN PROJECTS THAT INCLUDE THE VERTX-KAFKA-CLIENT ARTIFACT

In previous releases of Eclipse Vert.x, when building a project that includes the **vertx-kafka-client** artifact, Maven generated the following warning, potentially causing the build to fail:

> The POM for org.scala-lang:scala-compiler:jar:${starr.version} is missing, no dependency information available

This issue is resolved in the Eclipse Vert.x 3.9.6 release and no longer occurs.

5.3. VERT.X AMQP CLIENT:`AMQPRECEIVER` LOGS RECEIVED MESSAGES TO STANDARD OUTPUT AS EXPECTED

In previous releases of Eclipse Vert.x, a **messageHandler** attached to an **AmqpReceiver** instance that was set to print a message received from a **Sender** instance located on the same address to standard output would fail to print the message to the log despite having received it. The issue resulted from using a deprecated **createReceiver** method to initialize the **AmqpReceiver** instance. The deprecated method is removed from the **vertx-amqp-client** module in the Eclipse Vert.x 3.9.6 release and the issue no longer occurs.

CHAPTER 6. KNOWN ISSUES

6.1. RED HAT AMQ STREAMS IMAGES ARE NOT AVAILABLE FOR IBM Z AND IBM POWER SYSTEMS

The Red Hat AMQ Streams Operator and Kafka images are not available for IBM Z and IBM Power Systems. Since the images are not available, the **vertx-kafka-client** module is not certified to work with AMQ Streams on IBM Z and IBM Power Systems.

6.2. CONNECTION BETWEEN A RHEL 8-BASED DATABASE APPLICATION AND A RHEL 7-BASED MYSQL 5.7 DATABASE FAILS DUE TO TLS PROTOCOL VERSION MISMATCH

Description

Attempting to open a TLS-secured connection using OpenSSL between an application container built on a RHEL 8-based OpenJDK builder image and a database container built on a RHEL 7-based MySQL 5.7 container image results in a connection failure due to a **javax.net.ssl.SSLHandshakeException** at runtime: For more detail, view the issue in JIRA .

```
...
Caused by: javax.net.ssl.SSLHandshakeException: No appropriate protocol (protocol is disabled or
cipher suites are inappropriate)
...
```

Cause

The issue occurs due to a difference in the latest supported TLS protocol version between RHEL 7 and RHEL 8. The TLS implementation on RHEL 7 supports TLS protocol versions 1.0 (deprecated), 1.1, and 1.2. The TLS implementation on RHEL 8 also supports TLS protocol version 1.3, which is also the default TLS version used in RHEL 8-based builder images. This discrepancy may cause a TLS protocol version mismatch between application components while negotiating a TLS handshake, which in turn causes the connection between the application and database containers to fail.

Workaround

To prevent the issue described above, manually specify a TLS protocol version that is supported on both operating system versions in your database connection string. For example:

```
jdbc:mysql://testdb-mysql:3306/testdb?enabledTLSProtocols=TLSv1.2
```

6.3. FALSE CONNECTION RESET BY PEER ERROR MESSAGES WHEN CALLING APPLICATION ENDPOINT

Making an HTTP request on an endpoint of an Eclipse Vert.x application using either the **curl** tool or a Java HTTP client, produces the following error in the logs after each request:

```
io.vertx.core.net.impl.ConnectionBase
SEVERE: java.io.IOException: Connection reset by peer
```

This behavior is caused by the interaction of the Netty application framework and the HAProxy load-balancer used by OpenShift. The error occurs due to existing HTTP connections being re-used by

HAProxy without closing. Even though the error message is logged, no error condition occurs. HTTP requests are handled correctly and the application responds as expected.

CHAPTER 7. ADVISORIES RELATED TO THIS RELEASE

The following advisories have been issued to document enhancements, bugfixes, and CVE fixes included in this release.

- RHEA-2021:0709