Red Hat build of Eclipse Vert.x 3.9

Eclipse Vert.x3.9のリリースノート

Eclipse Vert.x3.9.6で使用する場合

Last Updated: 2021-03-23

Red Hat build of Eclipse Vert.x 3.9 Eclipse Vert.x3.9のリリースノート

Eclipse Vert.x3.9.6で使用する場合

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

このリリースノートには、Eclipse Vert.x3.9.6に関連する重要な情報が含まれています。

目次

巻頭言

リリースの日付: 2021-03-11

RED HAT ビルドの ECLIPSE VERT.X 3.9 - ライフサイクルの終了

Red Hat ビルドの Eclipse Vert.x 3.9 は、メジャーバージョン 3.x で最後にリリースされるサポート対象リリースです。Eclipse Vert.x 3.x バージョンの完全サポートライフサイクルは 2021 年 3 月 31 日に終了します。詳細は、製品ライフサイクル ページを参照してください。

アプリケーションを Red Hat ビルドの Eclipse Vert.x 4 に移行することが推奨されます。

詳細は、Eclipse Vert.x 4 の ドキュメント を参照してください。

Eclipse Vert.x 4.0 移行ガイド では、Eclipse Vert.x 3.x アプリケーションを Eclipse Vert.x 4 にアップグレードする方法を説明します。

RED HAT ドキュメントへのフィードバック

ご意見やご意見をお聞かせください。フィードバックを行うには、ドキュメント内のテキストを強調表示し、コメントを追加します。

本セクションでは、フィードバックの送信方法を説明します。

前提条件

- Red Hat カスタマーポータルにログインします。
- Red Hat カスタマーポータルで、**Multi-page HTML** 形式のドキュメントを表示します。

手順

フィードバックを提供するには、以下の手順を実施します。

1. ドキュメントの右上隅にある **Feedback** ボタンをクリックして、既存のフィードバックを表示します。

注記

フィードバック機能は、**Multi-page HTML** 形式でのみ有効です。

2. フィードバックを提供するドキュメントのセクションを強調表示します。
3. 強調表示されているテキストの近くに表示される **Add Feedback** ポップアップをクリックします。
   ページの右側のフィードバックセクションに、テキストボックスが表示されます。
4. テキストボックスにフィードバックを入力し、**Submit** をクリックします。
   ドキュメントの問題が作成されました。
5. この問題を確認するには、フィードバックビューで問題トラッカーをクリックします。

第1章 必要なインフラストラクチャーコンポーネントのバージョン

Red Hat は、サポート対象として明示的に指定されたコンポーネントを除き、以下のコンポーネントに対するサポートは提供しません。

コンポーネント名	Version
Maven	3.6.0
Fabric8 Maven プラグイン	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 以降
CDK[g]	3.11.0
git	2.0 以降
oc コマンドラインツール	3.11 以降[h]

[a] JRE はソースから Java アプリケーションをコンパイルするツールを提供しないため、完全な JDK インストールが必要です。

[b] Red Hat OpenJDK は、Red Hat によってサポートされます。

[c] Red Hat は、JDK の LTS リリースのみをサポートします。

[d] 実稼働環境のスタンドアロン RHEL に CNR をベースとしたアプリケーションをデプロイする場合。

[e] 実稼働環境のスタンドアロン RHEL に CNR をベースとしたアプリケーションをデプロイする場合。

[f] OCP は、Red Hat によってサポートされています。

[g] Red Hat によってサポートされています。

[h] CLI ツール **oc** のバージョンは、使用している OCP のバージョンに対応する必要があります。

第2章 サポートされる ECLIPSE VERT.X RUNTIME コンポーネント設定および統合

以下のリソースは、Eclipse Vert.x と Red Hat 製品のサポートされる設定および統合を定義します。

- 実稼働環境で Eclipse Vert.x との統合でサポートされる技術の一覧は、「Red Hat Runtimes Supported Configurations」を参照してください。
- Eclipse Vert.x ランタイムアーティファクトおよびそのバージョンの一覧は、コンポーネントの詳細ページ を参照してください。

第3章 機能

3.1. 新機能および変更された機能

本項では、本リリースで導入された新機能を説明します。また、既存の機能の変更に関する情報も含まれます。

3.1.1. EventBus JavaScript クライアントの変更点

以下のセクションでは、EventBus JavaScript クライアントの変更を説明します。

3.1.1.1. EventBus JavaScript クライアントは npm でのみ利用できます。

Eclipse Vert.x 3.9.5 より前のリリースでは、イベントバス JavaScript クライアントは Maven Central、NPM などのさまざまな場所で利用できました。

Eclipse Vert.x 3.9.5 以降、EventBus JavaScript クライアントモジュールは新しい npm の場所に利用できます。クライアントは他の場所で利用できません。以下の新しい 場所 からモジュールを使用するようにビルドシステムを更新する必要があります。

3.1.1.2. JavaScript クライアントのバージョン管理

Eclipse Vert.x 3.9.5 より前のリリースでは、すべての Eclipse Vert.x リリースには JavaScript クライアントの新しいリリースが含まれていました。

しかし、Eclipse Vert.x 3.9.5 以降では、クライアントに変更がある場合に限り、新しいバージョンの JavaScript クライアントを npm で利用できます。バージョン変更がなければ、すべての Eclipse Vert.x リリースでクライアントアプリケーションを更新する必要はありません。

3.1.2. **Row** および **Tuple** の値アクセサーメソッドでは、無効な値で例外が発生

Row および **Tuple** の値アクセサーメソッドは、無効な値で **null** を返す代わりに例外が発生

- 指定の列のルックアップが存在しない場合に **Row** のメソッドでは **NoSuchElementException** が発生
- **Row** および **Tuple** のメソッドでは、戻り値を予想される型にキャストできないか、または強制できない場合に **ClassCastException** が発生

3.1.3. 同じ **Verticle** の複数のインスタンスと共有されるワイルドカードポート (0)

同じ Verticle の複数のインスタンスがワイルドカードポート (0) を使用する場合は、最初のインスタンスはランダムポートにバインドされ、その他のインスタンスも同じランダムなポートにバインドします。

Eclipse Vert.x 3.9.2 より前のリリースでは、ワイルドカードポート (0) を使用する同じ Verticle のインスタンスはすべて、異なるランダムなポートにバインドされました。

3.1.4. **SqlConnection.Prepare** メソッドを使用して作成された準備済みステートメントはキャッシュされていません

SqlConnection.Prepare メソッドを使用して作成される準備済みステートメントは、内部でキャッシュされなくなりました。準備済みステートメントをキャッシュする場合は、キャッシュを設定する必要があります。

3.1.5. pgConnection.Prepare メソッドは名前付きの準備済みステートメントを作成します。

PgConnection.Prepare メソッドは、名前付きの準備済みステートメントを作成します。以前のリリースでは、メソッドにより、名前が付いていない準備済みステートメントが作成されていました。

3.1.6. IBM Z および IBM Power Systems での Eclipse Vert.x Runtime のサポート

s390x および ppc64le プラットフォーム向け Red Hat ビルドの Eclipse Vert.x は、IBM Z および IBM Power Systems インフラストラクチャーにプロビジョニングされる OpenShift 環境でのみサポートされています。IBM Z および IBM Power Systems での RHEL のスタンドアロンインストールでの Eclipse Vert.x アプリケーションの実行はサポートされていません。

IBM Z および IBM Power System でサポートされている Eclipse OpenJ9 Java イメージと、IBM Z および IBM Power System でサポートされている製品向け新しいイメージは、Red Hat Ecosystem Catalog から入手できます。

3.1.7. IBM Z および IBM Power System のインフラストラクチャーでプロビジョニングされた OpenShift でのサンプルアプリケーションのデプロイ

IBM Z および IBM Power System インフラストラクチャーでプロビジョニングされている OpenShift 環境でサンプルアプリケーションをデプロイするには、**pom.xml** ファイルおよびコマンドに関連する IBM Z および IBM Power System のイメージ名を指定します。

サンプルアプリケーションの一部には、ワークフローを実証するために Red Hat Data Grid などの他の製品も必要になります。この場合は、これらの製品のイメージ名を、サンプルアプリケーションの YAML ファイルで関連する IBM Z および IBM Power System のイメージ名に変更する必要もあります。

3.1.8. Fluent Query メソッド

新しいクラス Query は Fluent API です。これにより、実行前にクエリーを作成および設定できます。すべてのクエリーコレクターは **Query** インターフェースで利用できます。

新しいクラス **PreparedStatement** を使用すると、準備済みステートメントを作成および実行することができます。準備済みステートメントは、カーソル操作やストリーム操作などの複数の対話を実行できます。

既存のクラス **PreparedQuery** は、新しいクラス **Query** を拡張します。クラスは、接続のコンテキスト外で使用できるようになりました。

たとえば、以下のサンプルコードを使用してクエリーを作成します。

- コレクタークエリー

```
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 -> ...);
```

- 準備済みクエリー

```
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. 非推奨の機能

本リリースでは、以下の機能が非推奨になりました。

3.2.1. Mqtt クライアントオプションで keep alive time seconds メソッドが非推奨になる

MqttClientOptions.getKeepAliveTimeSeconds() メソッドおよび **MqttClientOptions.setKeepAliveTimeSeconds()** メソッドが非推奨になりました。**MqttClientOptions.getKeepAliveInterval()** メソッドおよび **MqttClientOptions.setKeepAliveInterval()** メソッドを使用して、秒単位でキープアライブの間隔を取得および設定します。

3.2.2. HTTP サーバーリクエストの net ソケットメソッドが非推奨に

HttpServerRequest.netSocket() メソッドが非推奨になりました。代わりに新しいメソッド **HttpServerRequest.toNetSocket()** を使用して、クライアントで TCP トンネルを確立します。

以下の例は、非推奨となったメソッド **HttpServerRequest.netSocket()** がどのように使用されているかを示しています。

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

以下の例は、新しいメソッド **HttpServerRequest.toNetSocket()** を使用する方法を示しています。

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

3.2.3. HTTP サーバーリクエストのアップグレードメソッドが非推奨に

HttpServerRequest.upgrade() メソッドが非推奨になりました。現在のリクエストの接続を WebSocket にアップグレードするには、代わりに **HttpServerResponse.toWebSocket()** を使用します。

以下の例は、非推奨となったメソッド **HttpServerRequest.upgrade()** がどのように使用されているかを示しています。

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

以下の例は、新しいメソッド **HttpServerResponse.toWebSocket()** を使用する方法を示しています。

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

3.2.4. sockjs.BridgeOptions クラスを非推奨に

sockjs.BridgeOptions クラスが非推奨になりました。代わりに新しい **sockjs.SockJSBridgeOptions** クラスを使用してください。**sockjs.SockJSBridgeOptions** クラスには、イベントバスブリッジの設定に必要なすべてのオプションが含まれます。

データオブジェクトクラスの名前が変更された場合を除き、新しいクラスの動作は変更されません。新規ブリッジが追加されると、重複した設定が多数ありました。新しいクラスには可能な共通設定がすべて含まれ、重複した設定を削除します。

3.2.5. Future パラメーターを使用した Verticle start および stop メソッドが非推奨に

verticle メソッドの **start(Future<Void>)** および **stop(Future<Void>)** が非推奨になりました。**start(Promise<Void>)** および **stop(Promise<Void>)** のメソッドを使用して、プロミスキャスで作業を行います。

3.2.6. DNS の拡張メカニズムがデフォルトで無効になっています。

DNS の拡張メカニズムは、Eclipse Vert.x ではデフォルトで無効になっています。EDNS を有効にするには、**AddressResolverOptions.setOptResourceEnabled()** メソッドを使用します。入力パラメーター **optResourceEnabled** を **true** に指定します。

3.2.7. 新しい接続ハンドラーメソッド

以下の新しいハンドラーメソッドが利用できます。

- **HttpClientRequest.connectionHandler()** メソッドは非推奨となり、今後のリリースで削除されます。代わりに **HttpClient.connectionHandler()** メソッドを使用して、アプリケーションのクライアントリクエストの接続ハンドラーを呼び出します。たとえば、以下のコードを使用して **HttpClient.connectionHandler()** メソッドと連携します。

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

- **Future<T>.setHandler()** メソッドは非推奨となり、今後のリリースで削除されます。代わりに、**Future<T>.onComplete()** メソッド、**Future<T>.onSuccess()** メソッド、および **Future<T>.onFailure()** メソッドを使用して、それぞれアクションの完了、成功、および失敗の結果でハンドラーを呼び出します。たとえば、**Future<T>.onComplete()** メソッドを使用するには、以下のコードを使用します。

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

3.2.8. AdminUtils クラスが利用できなくなる

AdminUtils クラスが利用できなくなりました。代わりに新しい **KafkaAdminClient** クラスを使用して、Kafka クラスターで管理操作を実行します。

3.2.9. WebSocket の HTTP メソッドの更新

WebSocket の HTTP メソッドの更新は次のとおりです。

- メソッド名で **WebSocket** という用語の使用に一貫性がありませんでした。メソッド名に、**WebSocket** ではなく **Websocket** などの誤った大文字が含まれていました 。以下のクラスで **Web ソケット** の使用に一貫性のないメソッドが非推奨となり、今後のリリースで削除されます。代わりに正しい大文字を持つ新しいメソッドを使用してください。
  - **HttpServerOptions** クラスの以下のメソッドは非推奨になりました。

非推奨となったメソッド	新しいメソッド
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()**

非推奨となったメソッド	新しいメソッド
getWebsocketSubProtocols()	**getWebSocketSubProtocols()**
setWebsocketSubProtocols()	**setWebSocketSubProtocols()**

WebSocket サブプロトコルの新しいメソッドは、項目を保存するためにコンマ区切りの文字列の代わりに **List<String>** データ型を使用します。

- **HttpClientOptions** クラスの以下のメソッドは非推奨になりました。

非推奨となったメソッド	新しいメソッド
getTryUsePerMessageWebsocketCompression()	**getTryUsePerMessageWebSocketCompression()**
setTryUsePerMessageWebsocketCompression()	**setTryUsePerMessageWebSocketCompression()**
getTryWebsocketDeflateFrameCompression()	**getTryWebSocketDeflateFrameCompression()**
getWebsocketCompressionAllowClientNoContext()	**getWebSocketCompressionAllowClientNoContext()**
setWebsocketCompressionAllowClientNoContext()	**setWebSocketCompressionAllowClientNoContext()**
getWebsocketCompressionLevel()	**getWebSocketCompressionLevel()**
setWebsocketCompressionLevel()	**setWebSocketCompressionLevel()**
getWebsocketCompressionRequestServerNoContext()	**getWebSocketCompressionRequestServerNoContext()**
setWebsocketCompressionRequestServerNoContext()	**setWebSocketCompressionRequestServerNoContext()**

- **HttpServer** クラスの以下のハンドラーメソッドが非推奨になりました。

非推奨となったメソッド	新しいメソッド
websocketHandler()	**webSocketHandler()**
websocketStream()	**webSocketStream()**

- **WebsocketRejectedException** は非推奨になりました。代わりに **UpgradeRejectedException** を出力します。

3.2.10. 非推奨になった認証および承認のクラスおよびメソッド

以下の認証および承認クラスおよびメソッドは非推奨となり、今後のリリースで置き換えられます。

- クラス:
  - **AbstractUser**
  - **PubSecKeyOptions**
  - **JDBCAuthOptions**
  - **JDBCHashStrategy**
  - **AccessToken**
  - **KeycloakHelper**
  - **ShiroAuth**
  - **AuthProviderInternal**
- メソッド:
  - **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. 共有データソースのないクライアントを作成するメソッド

次の新しいメソッドを使用して、他のクライアントとデータソースを共有していないクライアントを作成します。これらのメソッドは、独自のデータソースを維持します。

非推奨となったメソッド	新しいメソッド
MongoClient.createNonShared()	**MongoClient.create()**
JDBCClient.createNonShared()	**JDBCClient.create()**

非推奨となったメソッド	新しいメソッド
CassandraClient.createNonShared()	CassandraClient.create()
MailClient.createNonShared()	MailClient.create()

第4章 リリースコンポーネント

4.1. 本リリースで導入されたサポート対象のアーティファクト

本リリースでは、テクノロジープレビューからフルサポートになったアーティファクトはありません。

4.2. 本リリースで導入されたテクニカルプレビューアーティファクト

本リリースでは、新たにテクノロジープレビューとして提供されたアーティファクトはありません。

注記

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、「テクノロジープレビュー機能のサポート範囲」を参照してください。

4.3. 本リリースで削除されたアーティファクト

本リリースでは削除になったアーティファクトはありません。

4.4. 本リリースで非推奨となったアーティファクト

本リリースでは、非推奨となったアーティファクトはありません。

第5章 修正された問題

この Eclipse Vert.x リリースには、バージョン 3.9.6 のコミュニティーリリースのすべてのバグ修正が含まれます。コミュニティーリリースで解決された問題は、Eclipse Vert.x 3.9.6 コミュニティーのリリースノート に記載されています。

5.1. RXJAVA1 および RXJAVA2 の依存関係が POM ファイルにデフォルトで追加されます。

Eclipse Vert.x 3.9.5 では、**vertx-rx-java** モジュールおよび **vertx-rx-java2** モジュールは POM ファイルで任意の依存関係として表示されます。ただし、デフォルトでは、Eclipse Vert.x JUnit5 拡張コードは、Rx バージョンの **Vertx** クラスおよび **WebClient** クラスを読み込んでいました。これらの依存関係は利用できないため、例外が発生していました。

この問題は Eclipse Vert.x 3.9.6 で修正されました。RxJava1 および RxJava2 の依存関係が POM ファイルにデフォルトで追加されました。

5.2. STARR.VERSION プロパティーは、VERTX-KAFKA-CLIENT アーティファクトを含むプロジェクトに欠落していると表示されなくなりました。

以前のリリースの Eclipse Vert.x では、**vertx-kafka-client** アーティファクトを含むプロジェクトをビルドすると、Maven は以下の警告を生成して、ビルドが失敗する可能性がありました。

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

この問題は Eclipse Vert.x 3.9.6 リリースで解決され、発生しなくなりました。

5.3. VERT.X AMQP CLIENT:`AMQPRECEIVER` ログが想定どおりに標準出力に受信されました。

以前のリリースの Eclipse Vert.x では、標準出力と同じアドレスにある **Sender** インスタンスから受信したメッセージを出力するように **AmqpReceiver** インスタンスを設定し、それに **messageHandler** を登録し、メッセージを受信しても、ログに出力できませんでした。この問題は、非推奨の **createReceiver** メソッドを使用して **AmqpReceiver** インスタンスを初期化したために発生していました。非推奨のメソッドは Eclipse Vert.x 3.9.6 リリースの **vertx-amqp-client** モジュールから削除され、問題は発生しなくなりました。

第6章 既知の問題

6.1. IBM Z および IBM POWER SYSTEMS では、RED HAT AMQ STREAMS イメージが利用できない

Red Hat AMQ Streams Operator および Kafka イメージは、IBM Z および IBM Power Systems では利用できません。イメージは利用できないため、**vertx-kafka-client** モジュールは IBM Z および IBM Power Systems 上の AMQ Streams と動作することが認定されていません。

6.2. TLS プロトコルのバージョンが一致しないため、RHEL 8 ベースのデータベースアプリケーションと RHEL 7 ベースの MYSQL 5.7 データベース間の接続が失敗する

説明

RHEL 8 ベースの OpenJDK ビルダーイメージで構築されたアプリケーションコンテナーと、RHEL 7 ベースの MySQL 5.7 コンテナーイメージ上に構築されたデータベースコンテナーとの間で、OpenSSL を使用して TLS でセキュア化された接続を開くと、ランタイム時に **javax.net.ssl.SSLHandshakeException** によって接続に失敗します。詳細は、JIRA の問題 を参照してください。

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

原因

この問題は、RHEL 7 から RHEL 8 でサポートされる最新の TLS プロトコルバージョンが異なるために発生します。RHEL 7 の TLS 実装は、TLS プロトコルバージョン 1.0 (非推奨)、1.1、および 1.2 に対応しています。RHEL 8 の TLS 実装は、TLS プロトコルバージョン 1.3 にも対応しています。これは、RHEL 8 ベースのビルダーイメージで使用されるデフォルトの TLS バージョンでもあります。この不一致により、TLS ハンドシェイクのネゴシエーション中に、アプリケーションコンポーネント間で TLS プロトコルバージョンの不一致が発生し、アプリケーションとデータベースコンテナーとの間の接続が失敗する可能性があります。

回避策

上記の問題を防ぐには、データベース接続文字列の両方のオペレーティングシステムバージョンでサポートされる TLS プロトコルバージョンを手動で指定します。以下は例になります。

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

6.3. FALSE CONNECTION がアプリケーションエンドポイントを呼び出す際にピアエラーメッセージによってリセットされる

curl ツールまたは Java HTTP クライアントのいずれかを使用して Eclipse Vert.x アプリケーションのエンドポイントで HTTP リクエストを行うと、リクエストごとに以下のエラーがログに出力されます。

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

この動作は、Netty アプリケーションフレームワークと、OpenShift によって使用される HAProxy ロードバランサーの対話によって生じます。このエラーは、HAProxy が閉じずに既存の HTTP 接続が再使用されるため発生します。エラーメッセージがログに記録されても、エラー状態は発生しません。HTTP リクエストが正しく処理され、アプリケーションは予想通りに応答します。

第7章 本リリースに関連するアドバイザリー

本リリースに含まれるドキュメントの機能強化、バグ修正、および CVE の修正については、以下のアドバイザリー提供されています。

- RHEA-2021:0709