Red Hat build of OpenJDK 11

Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 への移行

Last Updated: 2024-05-10

Red Hat build of OpenJDK 11 Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 への移行

法律上の通知

概要

Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 への移行 ガイドでは、Red Hat build of OpenJDK 8 アプリケーションを Red Hat build of OpenJDK 11 にアップグレードする方法について説明します。

目次

RED HAT BUILD OF OPENJDK ドキュメントへのフィードバック

エラーを報告したり、ドキュメントを改善したりするには、Red Hat Jira アカウントにログインし、課題を送信してください。Red Hat Jira アカウントをお持ちでない場合は、アカウントを作成するように求められます。

手順

1. 次のリンクをクリックして **チケットを作成します**。
2. **Summary** に課題の簡単な説明を入力します。
3. **Description** に課題や機能拡張の詳細な説明を入力します。問題があるドキュメントのセクションへの URL を含めてください。
4. **Submit** をクリックすると、課題が作成され、適切なドキュメントチームに転送されます。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

第1章 OPENJDK 8 から OPENJDK 11 への移行の概要

Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 への移行ガイドでは、新機能や非推奨にされたり削除されたりした API など、Red Hat build of OpenJDK 8 からの移行に影響を与える可能性のある Red Hat build of OpenJDK 11 リリースの変更について説明します。このガイドの情報を使用して、Red Hat build of OpenJDK 8 の Java アプリケーションを Red Hat build of OpenJDK 11 にアップグレードできます。

OpenJDK プロジェクトは、更新を提供し、下位互換性を提供するための保守的なアプローチで知られています。ただし、プロジェクトの進化、セキュリティー、安定性を保証するために、Red Hat build of OpenJDK プロジェクトには、Red Hat build of OpenJDK のメジャーリリース間でいくつかの非互換性が生じる場合があります。これらの非互換性は、次のシナリオに関連しています。

- 廃止された、または安全でないと見なされる API を使用している場合。
- 実装の詳細であり、公開またはサポートされている API の詳細ではないとみなされるプロジェクトの内部にアクセスする場合。

1.1. RED HAT BUILD OF OPENJDK 8U および 11U について

OpenJDK は、Java Platform、Standard Edition (Java SE) の無料のオープンソースリファレンス実装です。Red Hat build of OpenJDK は、アップストリームの OpenJDK 8u、OpenJDK 11u、および OpenJDK 17u プロジェクトに基づいています。Shenandoah ガベージコレクターは、Red Hat build of OpenJDK のバージョン 8、11、および 17 に含まれています。

Red Hat build of OpenJDK には次の利点があります。

- **マルチプラットフォーム**: Red Hat build of OpenJDK が RHEL および Microsoft Windows でサポートされるようになったため、デスクトップ、データセンター、およびハイブリッドクラウド環境全体で単一の Java プラットフォーム上のアプリケーションを標準化できます。
- **頻繁なリリース**: Red Hat は、Red Hat build of OpenJDK 8、Red Hat build of OpenJDK 11、および Red Hat build of OpenJDK 17 ディストリビューションに対して、JRE および JDK の四半期の更新を提供します。これらの更新は、アーカイブ、RPM、および Windows MSI ベースのインストーラーファイルとコンテナーイメージとして利用できます。
- **長期サポート**: Red Hat は、最近リリースされた Red Hat build of OpenJDK 8、Red Hat build of OpenJDK 11、および Red Hat build of OpenJDK 17 ディストリビューションをサポートします。

関連情報

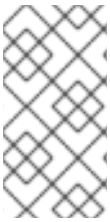
- サポートライフサイクルの詳細は、OpenJDK のライフサイクルおよびサポートポリシー を参照してください。

第2章 RED HAT BUILD OF OPENJDK 8 と RED HAT BUILD OF OPENJDK 11 の主な違い

Java アプリケーションを Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 に移行する前に、Red Hat build of OpenJDK 11 の更新と変更を理解する必要があります。これらの更新と変更により、アプリケーションを Red Hat build of OpenJDK 11 に移行する前に、Red Hat build of OpenJDK 8 を設定することが必要になる場合があります。

Red Hat build of OpenJDK 8 と Red Hat build of OpenJDK 11 の主な違いの 1 つは、Red Hat build of OpenJDK 11 にモジュールシステムが含まれていることです。Red Hat build of OpenJDK 8 アプリケーションを Red Hat build of OpenJDK 11 に移行する場合は、アプリケーションのライブラリーとモジュールを Red Hat build of OpenJDK 8 のクラスパスから Red Hat build of OpenJDK 11 のモジュールクラスに移動することを検討してください。この変更により、アプリケーションのクラスローディング機能を向上させることができます。

Red Hat build of OpenJDK 11 には、メモリー使用量の向上、起動速度の向上、コンテナー統合の向上など、アプリケーションのパフォーマンスを向上させることができる新機能と機能拡張が含まれています。



注記

一部の機能は、Red Hat build of OpenJDK と、OpenJDK の他のアップストリームバージョンまたはサードパーティーバージョンとで異なる場合があります。たとえば、Shenandoah ガベージコレクター (GC) はすべての Red Hat build of OpenJDK で使用できますが、他の builds of OpenJDK ではデフォルトで使用できない場合があります。

2.1. 暗号化とセキュリティー

Red Hat build of OpenJDK 8 と Red Hat build of OpenJDK 11 の間には、暗号化とセキュリティーに若干の違いがあります。ただし、Red Hat build of OpenJDK の両バージョンには、多くの類似した暗号化およびセキュリティー動作があります。

OpenJDK の Red Hat ビルドはシステム全体の証明書を使用し、各ビルドはシステムのグローバル設定から無効な暗号化アルゴリズムのリストを取得します。これらの設定は、Red Hat がサポートするすべての Red Hat build of OpenJDK リリースに共通であるため、Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 に簡単に変更できます。

FIPS モードでは、Red Hat build of OpenJDK 8 と Red Hat build of OpenJDK 11 のリリースは自己設定されるため、どちらのリリースでも起動時に同じセキュリティープロバイダーが使用されます。

Red Hat build of OpenJDK 8 と Red Hat build of OpenJDK 11 の TLS スタックは似ています。これは、Red Hat build of OpenJDK 11 の SunJSSE エンジンが Red Hat build of OpenJDK 8 にバックポートされたためです。Red Hat build of OpenJDK の両バージョンが、TLS 1.3 プロトコルをサポートしています。

Red Hat build of OpenJDK 8 と Red Hat build of OpenJDK 11 の間には、次のように、暗号化とセキュリティーに若干の違いがあります。

- Red Hat build of OpenJDK 8 では、クライアント側の TLSv1.3 がデフォルトで無効になっているため、Red Hat build of OpenJDK 8 の TLS クライアントは、ターゲットサーバーとの通信プロセス中に TLSv1.3 プロトコルを使用しません。**jdk.tls.client.protocols** システムプロパティーを使用して、Red Hat build of OpenJDK 11 に **-Djdk.tls.client.protocols=TLSv1.3** プロパティーを設定することで、この動作を変更できます。

- Red Hat build of OpenJDK 11 は、Diffie-Hellman 鍵交換での **X25519** および **X448** EC 曲線の使用をサポートしています。このサポートは、Red Hat build of OpenJDK 8 では利用できません。

- Red Hat build of OpenJDK 11 は、従来の KRB5 ベースの暗号スイートをサポートしていません。Red Hat build of OpenJDK 8 は引き続きレガシー KRB5 ベースの暗号スイートをサポートしますが、**jdk.tls.client.cipherSuites** および **jdk.tls.server.cipherSuites** システムプロパティーを変更して、これらの暗号スイートを有効にする必要があります。

- Red Hat build of OpenJDK 11 は Datagram Transport Layer Security (DTLS) プロトコルをサポートしているため、TLS クライアントとサーバーは DTLS の **max_fragment_length** エクステンションを使用できます。Red Hat build of OpenJDK 8 は、このプロトコルをサポートしていません。

2.2. ガベージコレクター (GC)

Red Hat build of OpenJDK 8 はデフォルトのガベージコレクターとして Parallel GC を使用しますが、Red Hat build of OpenJDK 11 はデフォルトのガベージコレクターとして G1 GC を使用します。ガベージコレクターを選択する前に、次の詳細を考慮してください。

- スループットを向上させたい場合は、Parallel GC を使用してください。Parallel GC はスループットを最大化しますが、そのせいで時折 **stop-the-world** コレクションが一時停止することがあります。

- スループットとレイテンシーのバランスをとるためには G1 GC を使用してください。この GC は、**100** ミリ秒の範囲の一時停止時間でレイテンシーを実行できます。アプリケーションを Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 に移行するときにスループットの問題に気付いた場合は、Parallel GC に切り替えることができます。

- 低レイテンシーのコレクションには Shenandoah GC を使用してください。

Parallel GC を選択するには、-**XX:+UseParallelGC** などの **-XX:+<gc_type>** JVM オプションを使用します。

2.3. ガベージコレクター (GC) のロギングオプション

Red Hat build of OpenJDK 11 には、古いロギングフレームワークと比較してより効果的に機能する新しいロギングフレームワークが含まれています。Red Hat build of OpenJDK 11 には、統合された JVM ロギングオプションと統合された GC ロギングオプションも含まれています。

Red Hat build of OpenJDK 11 のロギングシステムは、デフォルトで - **XX:+PrintGCTimeStamps** および **-XX:+PrintGCDateStamps** オプションをアクティブにします。Red Hat build of OpenJDK 11 のロギング形式は Red Hat build of OpenJDK 8 とは異なるため、GC ログを解析するコードを更新する必要がある場合があります。

Red Hat build of OpenJDK 11 では、新しいフレームワークオプションのエイリアスを介して、古いロギングフレームワークオプションに引き続きアクセスできます。Red Hat build of OpenJDK 11 をより効率的に使用したい場合は、新しいロギングフレームワークオプションを使用してください。

Red Hat build of OpenJDK 11 では、ロギングフレームワークから次の Red Hat build of OpenJDK 8 オプションが置き換えられるか、削除されました。

Red Hat build of OpenJDK 8 のオプション	Red Hat build of OpenJDK 11 のオプション
-verbose:gc	**-Xlog:gc**
-XX:+PrintGC	**-Xlog:gc**
-XX:+PrintGCDetails	**-Xlog:gc***
-Xloggc	**-Xlog:gc:file=<path_to_filename>**

-XX:+ PrintGCDetails オプションを使用する場合は、**-Xlog:gc*** フラグを渡します。* は、より詳細なロギングをアクティブにします。

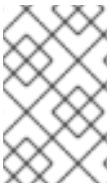
-Xloggc を使用する場合は、**:file=<filename>** オプションを追加して、ログ出力を指定されたファイルにリダイレクトします。たとえば、**-Xlog:gc:file=<filename>** です。

注記

Java HotSpot 仮想マシンで古いタグオプションを指定すると、仮想マシンは使用可能な新しいタグオプションの入力を求めます。古いタグオプションまたは新しいタグオプションのいずれかを使用することを選択できます。

Red Hat build of OpenJDK 11 には、次のオプションは含まれていません。Red Hat build of OpenJDK 11 でいずれかのオプションを使用しようとすると、起動エラーが発生します。

- **-Xincgc**
- **-XX:+CMSIncrementalMode**
- **-XX:+UseCMSCompactAtFullCollection**
- **-XX:+CMSFullGCsBeforeCompaction**
- **-XX:+UseCMSCollectionPassing**



注記

Red Hat build of OpenJDK 11 で前述のオプションのいずれかを使用する場合は、コマンドラインインターフェイスで **-XX:+IgnoreUnrecognizedVMOptions** オプションを発行して、起動の問題を無視できます。

関連情報

- 統合 JVM ロギングの一般的なフレームワークと **Xlog** オプションの形式の詳細については、JEP 158: Unified JVM Logging を参照してください。
- 削除されたオプションの詳細については、JEP 214: Remove GC Combinations Deprecated in JDK 8 を参照してください。
- 統合 GC ロギングの詳細については、JEP 271: Unified GC Logging を参照してください。

2.4. OPENJDK グラフィック

Red Hat build of OpenJDK 8 の Pisces とは対照的に、Red Hat build of OpenJDK 11 はデフォルトのレンダリングエンジンとして Marlin を使用します。Marlin レンダリングエンジンは、大量のアプリケーショングラフィックスの処理を改善します。

Marlin エンジンは Red Hat build of OpenJDK 8 で使用可能であり、有効にするには Java ランタイムで **-Dsun.java2d.renderer=sun.java2d.marlin.MarlinRenderingEngine** を発行します。

2.5. WEBSTART とアプレット

IcedTea-Web プラグインを使用すると、RHEL 7、RHEL 8、および Microsoft Windows オペレーティングシステム上の Red Hat build of OpenJDK 8 で Java WebStart を使用できます。Red Hat build of OpenJDK 11 以降のバージョンの Red Hat build of OpenJDK は、Java Webstart をサポートしていません。

アプレットは、Red Hat build of OpenJDK ではサポートされていません。

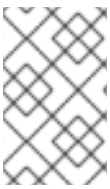
2.6. JAVA ライブラリークラス

Red Hat build of OpenJDK 9 で導入された Java Platform Module System (JPMS) は、非公開 API へのアクセスを制限または防止します。JPMS は、クラスパスやモジュールパスの使用など、Java アプリケーションを起動およびコンパイルする方法にも影響を与えます。

デフォルトでは、Red Hat build of OpenJDK 11 は JDK 内部モジュールへのアクセスを制限します。アプリケーションを移植する際のこの制限の回避策として、**java** コマンドにオプションを渡すことにより、アプリケーションの内部パッケージへのアクセスを提供できます。これらのオプションは、次の例で示されています。

```
--add-opens <module-name>/<package-in-module>=ALL-UNNAMED

--add-opens java.base/jdk.internal.math=ALL-UNNAMED
```



注記

Red Hat build of OpenJDK リリースサイクルの更新のため、以前の回避策は無期限に機能するわけではありません。Red Hat build of OpenJDK の将来のリリースでは、この回避策は廃止されます。

さらに、**--illegal-access=warn** オプションを **java** コマンドに渡すことで、不正アクセスのケースを確認できます。このオプションは、Red Hat build of OpenJDK のデフォルトの動作を変更します。

JPMS リファクターリングにより、Red Hat build of OpenJDK 11 の **ClassLoader** 階層が変更されます。Red Hat build of OpenJDK 11 は、システムクラスローダーを内部クラスに設定するため、Java プログラムはシステムクラスローダーにアクセスできません。

たとえば、**ClassLoader::getSystemClassLoader** を呼び出し、その結果を Red Hat build of OpenJDK 11 上の **URLClassLoader** にキャストする既存の Red Hat build of OpenJDK 8 のアプリケーションコードは、**URLClassLoader** がシステムクラスローダーの一部ではないため、機能しない可能性があります。Red Hat build of OpenJDK 11 で **URLClassLoader** に結果をキャストしようとすると、ランタイム例外メッセージを受け取る可能性があります。

Red Hat build of OpenJDK 11 には、特定のクラスのロードを制御できる新しいクラスローダーが含まれています。これにより、クラスローダーが必要なすべてのクラスを見つける方法が改善されます。

クラスローダーを作成すると、その親として **null** を渡すことができなくなります。Red Hat build of

OpenJDK 11 でこれを行うと、親としてクラスローダーが選択されず、新しいクラスローダーはプラットフォームクラスを見つけることができません。**ClassLoader.getPlatformClassLoader()** API を使用して、プラットフォームクラスローダーのインスタンスを取得できます。

関連情報

- JPMS の詳細については、JEP 261: Module System を参照してください。

2.7. 拡張機能および推奨オーバーライドメカニズム

Red Hat build of OpenJDK 11 では、オプションのパッケージをサポートしていた拡張メカニズムが利用できなくなりました。Red Hat build of OpenJDK 11 では、推奨標準オーバーライドメカニズムも削除されました。

<JAVA_HOME>/lib/ext または **<JAVA_HOME>/lib/endorsed** に追加されたライブラリーは使用できません。Red Hat build of OpenJDK 11 は、これらのディレクトリーを検出するとエラーを生成します。Red Hat build of OpenJDK 11 でオプションパッケージを使用する場合は、**jlink** ツールを使用してオプションパッケージのモジュールを作成し、そのツールを使用して、作成したモジュールにカスタムランタイムを含めることができます。

関連情報

- 削除されたメカニズムの詳細については、JEP 220: Modular Run-Time Images を参照してください。

2.8. OPENJDK 11 から非推奨化および削除された機能

Red Hat build of OpenJDK 8 でサポートされている一部の機能は、Red Hat build of OpenJDK 11 では非推奨になったか、削除されました。

COBRA

Red Hat build of OpenJDK 11 は次のツールをサポートしていません。

- **ldlj**
- **orbd**
- **servertool**
- **tnamesrv**

ロギングフレームワーク

Red Hat build of OpenJDK 11 は次の API をサポートしていません。

- **java.util.logging.LogManager.addPropertyChangeListener**
- **java.util.logging.LogManager.removePropertyChangeListener**
- **java.util.jar.Pack200.Packer.addPropertyChangeListener**
- **java.util.jar.Pack200.Packer.removePropertyChangeListener**
- **java.util.jar.Pack200.Unpacker.addPropertyChangeListener**
- **java.util.jar.Pack200.Unpacker.removePropertyChangeListener**

Java EE モジュール

Red Hat build of OpenJDK 11 は次の API をサポートしていません。

- **java.activation**
- **java.corba**
- **java.se.ee (aggregator)**
- **java.transaction**
- **java.xml.bind**
- **java.xml.ws**
- **java.xml.ws.annotation**

注記

外部依存関係は、前述の API の一部を提供する可能性があるため、依存関係によって提供された場合は、これらの API の使用を検討してください。

java.awt.peer

Red Hat build of OpenJDK 11 では、**java.awt.peer** パッケージが内部として設定されます。これは、アプリケーションがデフォルトでパッケージに自動的にアクセスできないことを意味します。この変更により、Red Hat build of OpenJDK 11 は、**Component.getPeer** メソッドなどのピア API を参照するクラスとメソッドを削除しました。
ユースケースの例は、ピア API を使用して次の基準をチェックすることです。

- コンポーネントを表示できる。
- コンポーネントは軽量コンポーネントである。
- コンポーネントは、OS ネイティブ UI コンポーネントによってサポートされている。

このコードは、**Component.isDisplayable()** と **Component.isLightweight()** を呼び出して更新することで、同じタスクを実行できます。

javax.security と java.lang API

Red Hat build of OpenJDK 11 は次の API をサポートしていません。

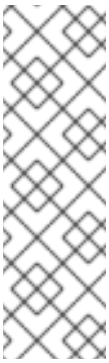
- **javax.security.auth.Policy**
- **java.lang.Runtime.runFinalizersOnExit(boolean)**
- **java.lang.SecurityManager.checkAwtEventQueueAccess()**
- **java.lang.SecurityManager.checkMemberAccess(java.lang.Class,int)**
- **java.lang.SecurityManager.checkSystemClipboardAccess()**
- **java.lang.SecurityManager.checkTopLevelWindow(java.lang.Object)**
- **java.lang.System.runFinalizersOnExit(boolean)**

- **java.lang.Thread.destroy()**
- **java.lang.Thread.stop(java.lang.Throwable)**

Sun.misc

sun.misc package は内部にあり、サポートされていません。Red Hat build of OpenJDK 11 では、次のパッケージが非推奨化または削除されました。

- **sun.misc.BASE64Encoder**
- **sun.misc.BASE64Decoder**
- **sun.reflect.Reflection**
  Red Hat build of OpenJDK 11 では、いくつかのメソッドが **sun.misc.Unsafe** パッケージから削除されました。



注記

可能な限り、**sun.misc package** コンポーネントの代わりにパブリック API を使用してください。たとえば、**sun.misc.Unsafe** パッケージの代わりに **VarHandles** パッケージを使用するか、**sun.reflect.Reflection** API の代わりに **StackWalker** API を使用します。

Red Hat build of OpenJDK 11 では、アプリケーションによる **sun.misc** パッケージ内のクラスの使用が制限されています。この問題の回避策の詳細については、Java library classes を参照してください。

関連情報

- Red Hat build of OpenJDK 8 の機能の詳細は、JDK 8 Features を参照してください。
- Red Hat build of OpenJDK 11 の機能の詳細は、JDK 11 を参照してください。
- 使用可能なすべての JEP のリストの詳細については、JEP 0: JEP Index を参照してください。
- 削除された Java EE モジュールと COBRA モジュール、およびこれらのモジュールの代替の可能性に関する詳細は、JEP 320: Remove the Java EE and CORBA Modules を参照してください。

第3章 移行の準備

Red Hat build of OpenJDK 11 には、Red Hat build of OpenJDK 8 にすでに正常にデプロイされているアプリケーションの再設定が必要になる可能性のある更新と変更が含まれています。

Red Hat build of OpenJDK 8 と Red Hat build of OpenJDK 11 の主な違いのセクションを確認し、相違点を移行計画に統合することで、効果的な移行計画を確実に実行できます。

Red Hat は、移行タスクを支援するために使用できる以下のツールを提供します。

- Migration Toolkit for Applications (MTA) は、Java アプリケーションを Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 に移行するために使用できる特定のツールです。

関連情報

- Red Hat build of OpenJDK リリース間の主な違いの詳細は、Red Hat build of OpenJDK 8 と Red Hat build of OpenJDK 11 の主な違い を参照してください。
- RHEL での Red Hat build of OpenJDK 11 のインストールの詳細は、**RHEL での Red Hat build of OpenJDK 11 のインストールと使用** ガイドを参照してください。
- Microsoft Windows での Red Hat build of OpenJDK 11 のインストールの詳細は、**Microsoft Windows 向けの Red Hat build of OpenJDK 8 のインストールと使用** ガイドを参照してください。
- RHEL での Red Hat build of OpenJDK バージョンの切り替えの詳細は、**RHEL での Red Hat build of OpenJDK 11 の設定** ガイドの RHEL のシステム全体の Red Hat build of OpenJDK バージョンを対話的に選択する を参照してください。
- Microsoft Windows での Red Hat build of OpenJDK バージョンの切り替えの詳細は、**Microsoft Windows での Red Hat build of OpenJDK 11 の設定** ガイドの インストールされているアプリケーションのバージョンから特定の Red Hat build of OpenJDK を選択する を参照してください。
- MTA ツールの詳細については、**Introduction to the Migration Toolkit for Applications**ガイドを参照してください。

第4章 アプリケーション移行用のツール

アプリケーションを Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 に移行する前に、ツールを使用して、Red Hat build of OpenJDK 11 で実行するアプリケーションの適合性をテストできます。

次の手順を使用して、テストプロセスを強化できます。

- サードパーティーのライブラリーを更新します。
- アプリケーションコードをコンパイルします。
- アプリケーションのコードで **jdeps** を実行します。
- Migration Toolkit for Applications (MTA) を使用して、Java アプリケーションを Red Hat build of OpenJDK 8 から Red Hat build of OpenJDK 11 に移行します。

関連情報

- MTA ツールの詳細については、Introduction to the Migration Toolkit for Applicationsガイドを参照してください。