



OpenJDK 17

OpenJDK 17.0.5 リリースノート

リリースノート

OpenJDK 17 OpenJDK 17.0.5 リリースノート

リリースノート

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

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Release_notes_for_OpenJDK_17.0.5.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

OpenJDK 17.0.5 ドキュメントのリリースノートには、OpenJDK 17 の新機能の概要と、潜在的な既知の問題と考えられる回避策のリストが記載されています。

目次

はじめに	3
多様性を受け入れるオープンソースの強化	4
RED HAT ドキュメントへのフィードバック (英語のみ)	5
第1章 OPENJDK のサポートポリシー	6
第2章 アップストリームの OPENJDK 17 との相違点	7
第3章 OPENJDK の機能	8
OpenJDK の機能強化	8
cpu.shares パラメーターが無効になっている	8
SHA-1 署名 JAR	8
SunMSCAPI プロバイダーは、新しい Microsoft Windows キーストアタイプをサポートします	9
HTTPURLConnection の keep-alive 動作を制御するためのシステムプロパティ	10
第4章 このリリースに関連するアドバイザリー	11

はじめに

OpenJDK (Open Java Development Kit) は、Java Platform Standard Edition (Java SE) のオープンソース実装です。OpenJDK の Red Hat ビルドは、OpenJDK 8u、OpenJDK 11u と OpenJDK 17u の 3 つのバージョンで利用できます。

Red Hat ビルドの OpenJDK 向けパッケージは、Red Hat Enterprise Linux および Microsoft Windows で利用でき、Red Hat Ecosystem Catalog の JDK および JRE として同梱されています。

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。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。フィードバックをお寄せいただくには、ドキュメントのテキストを強調表示し、コメントを追加できます。

このセクションでは、フィードバックの送信方法を説明します。

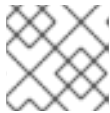
前提条件

- Red Hat カスタマーポータルにログインしている。
- Red Hat カスタマーポータルで、**マルチページ HTML** 形式でドキュメントを表示している。

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確認します。



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. 強調表示されたテキスト近くに表示される **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。ドキュメントに関する問題が作成されます。
5. 問題を表示するには、フィードバックビューで問題トラッカーリンクをクリックします。

第1章 OPENJDK のサポートポリシー

Red Hat は、一部の OpenJDK のメジャーバージョンをサポートします。一貫性を保つために、これらのバージョンは長期サポート (LTS) として指定されている Oracle JDK バージョンと同様のままとなります。

Red Hat は、Red Hat が OpenJDK を初めて導入してから 6 年間、OpenJDK のメジャーバージョンに対応しています。

OpenJDK 17 は、2027 年 11 月まで、Microsoft Windows および Red Hat Enterprise Linux で対応しています。



注記

RHEL 6 のライフサイクルは 2020 年 11 月に終了します。このため、OpenJDK はサポート対象設定として RHEL 6 をサポートしません。

関連情報

[OpenJDK Life Cycle and Support Policy \(Red Hat Customer Portal\)](#) を参照してください。

第2章 アップストリームの OPENJDK 17 との相違点

Red Hat Enterprise Linux の OpenJDK には、OpenJDK のアップストリームディストリビューションの構造上の変更が数多く含まれています。Microsoft Windows バージョンの OpenJDK は、Red Hat Enterprise Linux の更新にできる限り従います。

以下は、Red Hat OpenJDK 17 における最も注目すべき変更の一覧です。

- FIPS のサポート。Red Hat OpenJDK 17 は、RHEL が FIPS モードであるかどうかを自動的に検出し、そのモードで動作するように OpenJDK 17 を自動的に設定します。この変更は、Microsoft Windows 向けの OpenJDK ビルドには適用されません。
- 暗号化ポリシーのサポート。Red Hat OpenJDK 17 は、有効な暗号化アルゴリズムとキーサイズ制約のリストを RHEL システム設定から取得します。これらの設定コンポーネントは、トランスポート層セキュリティ (TLS) 暗号化プロトコル、証明書パス検証、および署名された JAR によって使用されます。さまざまなセキュリティプロファイルを設定して、安全性と互換性のバランスをとることができます。この変更は、Microsoft Windows 向けの OpenJDK ビルドには適用されません。
- Red Hat OpenJDK on RHEL は、アーカイブ形式のサポート用の **zlib**、イメージのサポート用の **libjpeg-turbo**、**libpng**、**giflib** などのネイティブライブラリーと動的にリンクします。また、RHEL はフォントのレンダリングと管理のために、**Harfbuzz** および **Freetype** に対して動的にリンクします。この変更は、Microsoft Windows 向けの OpenJDK ビルドには適用されません。
- **src.zip** ファイルには、OpenJDK に同梱されるすべての JAR ライブラリーのソースが含まれます。
- Red Hat OpenJDK on RHEL は、タイムゾーン情報のソースとして、システム全体のタイムゾーンデータファイルを使用します。
- Red Hat OpenJDK on RHEL は、システム全体の CA 証明書を使用します。
- Red Hat OpenJDK on Microsoft Windows には、RHEL で利用可能な最新のタイムゾーンデータが含まれています。
- Red Hat OpenJDK on Microsoft Windows は、RHEL から入手可能な最新の CA 証明書を使用します。

関連情報

- [Improve system FIPS detection \(RHEL Planning Jira\)](#) を参照してください。
- [Using system-wide cryptographic policies \(RHEL documentation\)](#) を参照してください。

第3章 OPENJDK の機能

最新の OpenJDK 17 リリースには、新機能が含まれている可能性があります。さらに、最新リリースは、以前の OpenJDK 17 リリースに由来する機能を強化、非推奨、または削除する可能性があります。



注記

その他の変更点やセキュリティ修正については、[OpenJDK 17.0.5 Released](#) を参照してください。

OpenJDK の機能強化

OpenJDK 17 では、以前のリリースの OpenJDK で作成された機能に拡張が行われました。

cpu.shares パラメーターが無効になっている

OpenJDK 17.0.5 リリースより前は、OpenJDK は、**cgroups** と呼ばれる Linux コントロールグループに属する **cpu.shares** パラメーターの誤った解釈を使用していました。このパラメーターにより、Java 仮想マシン (JVM) が使用可能な CPU よりも少ない CPU を使用する可能性があり、コンテナ内で動作するときの JVM の CPU リソースとパフォーマンスに影響を与える可能性があります。

OpenJDK 17.0.5 リリースでは、スレッドプールのスレッド数を決定するときに **cpu.shares** パラメーターを使用しないように JVM が設定されます。この設定を元に戻したい場合は、JVM の起動時に **-XX:+UseContainerCpuShares** 引数を渡します。



注記

-XX:+UseContainerCpuShares 引数は非推奨の機能であり、将来の OpenJDK リリースで削除される可能性があります。

[JDK-8281181](#) (JDK バグシステム) を参照してください。

SHA-1 署名 JAR

OpenJDK 17.0.5 リリースでは、**SHA-1** アルゴリズムで署名された JAR はデフォルトで制限され、署名されていないかのように扱われます。これらの制限は、次のアルゴリズムに適用されます。

- ダイジェスト、署名、およびオプションで JAR のタイムスタンプに使用されるアルゴリズム。
- コード署名者とタイムスタンプ機関の証明書チェーン内の証明書の署名アルゴリズムとダイジェストアルゴリズム、およびそれらの証明書が失効しているかどうかを確認するために使用される証明書失効リスト (CRL) またはオンライン証明書ステータスプロトコル (OCSP) 応答。

さらに、制限は署名済みの Java Cryptography Extension (JCE) プロバイダーにも適用されます。

以前にタイムスタンプが付けられた JAR の互換性リスクを軽減するために、この制限は、**SHA-1** アルゴリズムで署名され、**January 01, 2019** より前にタイムスタンプが付けられた JAR には適用されません。この例外は、将来の OpenJDK リリースで削除される可能性があります。

JAR ファイルが制限の影響を受けるかどうかを判断するには、CLI で次のコマンドを発行します。

```
$ jarsigner -verify -verbose -certs
```

前のコマンドの出力から、**SHA1**、**SHA-1**、または **disabled** のインスタンスを検索します。さらに、JAR が署名なしとして扱われることを示す警告メッセージを検索します。以下に例を示します。

Signed by "CN="Signer""
 Digest algorithm: SHA-1 (disabled)
 Signature algorithm: SHA1withRSA (disabled), 2048-bit key

WARNING: The jar will be treated as unsigned, because it is signed with a weak algorithm that is now disabled by the security property:

jdk.jar.disabledAlgorithms=MD2, MD5, RSA keySize < 1024, DSA keySize < 1024, SHA1 denyAfter 2019-01-01

新しい制限の影響を受けるすべての JAR をより強力なアルゴリズムに置き換えるか、再署名することを検討してください。

JAR ファイルがこの制限の影響を受ける場合は、アルゴリズムを削除して、**SHA-256** などのより強力なアルゴリズムでファイルに再署名できます。OpenJDK 17.0.5 の **SHA-1** 署名付き JAR に対する制限を削除する必要がある、セキュリティリスクを受け入れる場合は、次のアクションを実行できます。

1. **java.security** 設定ファイルを変更します。または、このファイルを保存して、必要な設定で別のファイルを作成することもできます。
2. **SHA1 usage SignedJAR & denyAfter 2019 01 011** エントリーを **jdk.certpath.disabledAlgorithms** セキュリティープロパティーから削除します。
3. **jdk.jar.disabledAlgorithms** セキュリティープロパティーから **SHA1 denyAfter 2019-01-01** エントリーを削除します。

注記

java.security ファイルの **jdk.certpath.disabledAlgorithms** の値は、RHEL 8 および 9 のシステムセキュリティポリシーによって上書きされる場合があります。システムセキュリティポリシーで使用される値は、ファイル **/etc/crypto-policies/back-ends/java.config** で確認でき、**java.security** ファイルで **security.useSystemPropertiesFile** を **false** に設定するか、**-Djava.security.disableSystemPropertiesFile=true** を JVM 渡すことで無効にします。これらの値はこのリリースでは変更されていないため、値は OpenJDK の以前のリリースと同じままです。

java.security ファイルの設定例については、JBoss EAP for OpenShift の **java.security** プロパティーのオーバーライド (Red Hat カスタマーポータル) を参照してください。

[JDK-8269039](#) (JDK バグシステム) を参照してください。

SunMSCAPI プロバイダーは、新しい Microsoft Windows キーストアタイプをサポートします

SunMSCAPI プロバイダーは、ローカル名前空間を **Windows-** に追加する必要がある次の Microsoft Windows キーストアタイプをサポートしています。

- **Windows-<local_computer_name>**
- **Windows-<root_local_computer_name>**
- **Windows-<current_username>**
- **Windows-<root_username>**

これらのタイプのいずれかを指定することにより、ローカルコンピューターの Microsoft Windows キーストアの場所へのアクセスを提供できます。これにより、ローカルシステムに保存されている証明書へのキーストアアクセスが提供されます。

[JDK-6782021](#) (JDK バグシステム) を参照してください。

HTTPURLConnection の keep-alive 動作を制御するためのシステムプロパティー

OpenJDK 17.0.5 リリースには、**HTTPURLConnection** の **keep-alive** 動作を制御するために使用できる次の新しいシステムプロパティーが含まれています。

- サーバーへの接続を制御する **http.keepAlive.time.server**。
- プロキシへの接続を制御する **http.keepAlive.time.proxy**。

OpenJDK 17.0.5 リリースより前では、**keep-alive** 時間が指定されていないサーバーまたはプロキシにより、ハードコーディングされたデフォルト値によって定義された期間、アイドル接続が開いたままになる場合があります。

OpenJDK 17.0.5 では、システムプロパティーを使用して **keep-alive** 時間のデフォルト値を変更できます。**keep-alive** プロパティーは、サーバーまたはプロキシのいずれかの HTTP **keep-alive** 時間を変更することでこの動作を制御します。これにより、OpenJDK の HTTP プロトコルハンドラーは、指定された秒数が経過した後にアイドル状態の接続を閉じます。

OpenJDK 17.0.5 リリースより前では、次の使用例は、**HTTPURLConnection** の特定の **keep-alive** 動作につながります。

- サーバーが **Connection:keep-alive** ヘッダーを指定し、サーバーの応答に **Keep-alive:timeout=N** が含まれている場合、クライアントの OpenJDK **keep-alive** キャッシュは **N** 秒のタイムアウトを使用します (**N** は整数値)。
- サーバーが **Connection:keep-alive** ヘッダーを指定しているが、サーバーの応答に **Keep-alive:timeout=N** のエントリーが含まれていない場合、クライアントの OpenJDK **keep-alive** キャッシュはプロキシに対して **60** 秒のタイムアウトを使用し、**5** サーバーの秒。
- サーバーが **Connection:keep-alive** ヘッダーを指定しない場合、クライアントの OpenJDK **keep-alive** キャッシュは、すべての接続に対して **5** 秒のタイムアウトを使用します。

OpenJDK 17.0.5 リリースでは、前述の動作が維持されていますが、2 番目と 3 番目に挙げた使用例におけるタイムアウトは、デフォルトの設定に依存するのではなく、**http.keepAlive.time.server** および **http.keepAlive.time.proxy** プロパティーを使用して指定できるようになっています。



注記

keep-alive プロパティーを設定し、サーバーが **Keep-Alive** 応答ヘッダーの **keep-alive** 時間を指定した場合、HTTP プロトコルハンドラーはサーバーによって指定された時間を使用します。この状況は、プロキシと同じです。

[JDK-8278067](#) (JDK バグシステム) を参照してください。

第4章 このリリースに関連するアドバイザリー

以下のアドバイザリーは、本リリースに含まれるバグ修正および CVE の修正に発行されています。

- [RHSA-2022:6999](#)
- [RHSA-2022:7000](#)
- [RHSA-2022:7001](#)
- [RHSA-2022:7051](#)
- [RHSA-2022:7054](#)

改訂日時: 2022-11-05 19:57:09 +1000