Red Hat build of OpenJDK 11

FIPS を使用した RHEL での Red Hat build of OpenJDK 17 の設定

Last Updated: 2024-05-10

Red Hat build of OpenJDK 11 FIPS を使用した RHEL での Red Hat build of OpenJDK 17 の設定

法律上の通知

概要

Red Hat build of OpenJDK は、Red Hat Enterprise Linux プラットフォーム上の Red Hat 製品です。FIPS を使用した RHEL での Red Hat build of OpenJDK 17 の設定 では、FIPS の概要と、FIPS で Red Hat build of OpenJDK を有効化および設定する方法を説明します。

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 FIPS (FEDERAL INFORMATION PROCESSING STANDARD) の概要

FIPS (Federal Information Processing Standards) は、コンピューターシステムやネットワーク間のセキュリティーおよび相互運用性を強化するためのガイドラインと要件を提供します。FIPS 140-2 および 140-3 シリーズは、ハードウェアおよびソフトウェアの両レベルで暗号化モジュールに適用されます。アメリカ国立標準技術研究所 は、進行中の暗号モジュールと承認済みの暗号モジュールの両方の検索可能なリストとともに 暗号化モジュール検証プロウラム を実装しています。

Red Hat Enterprise Linux (RHEL) は、FIPS 140-2 コンプライアンスシステム全体を有効にする統合フレームワークを提供します。FIPS モードで操作する場合、暗号化ライブラリーを使用するソフトウェアパッケージはグローバルポリシーに従って自己設定されます。ほとんどのパッケージでは、互換性やその他のニーズにおいて、デフォルトの調整動作を変更する手段を提供します。

Red Hat build of OpenJDK 17 は、FIPS ポリシー対応パッケージです。

関連情報

- FIPS モードを有効にして RHEL をインストールする方法は、FIPS モードが有効になっている RHEL 8 システムのインストール を参照してください。
- RHEL をインストールした後に FIPS モードを有効にする方法は、FIPS モードへのシステムの切り替え を参照してください。
- RHEL の FIPS モードで Red Hat build of OpenJDK を実行する方法は、Running OpenJDK in FIPS mode on RHEL を参照してください。
- Government 標準による Red Hat コンプライアンスの詳細は、Goverment Standards を参照してください。

第2章 FIPS モードでの OPENJDK 11 の RED HAT ビルドの設定

Red Hat build of OpenJDK 17 は、起動時に FIPS モードがシステムで有効になっているかどうかを確認します。yes の場合は、グローバルポリシーに従って FIPS を自己設定します。これは、RHEL 8.3 以降のデフォルトの動作です。以前の RHEL 8 リリースでは、**com.redhat.fips** システムプロパティーを JVM 引数として **true** に設定する必要があります。たとえば、**-Dcom.redhat.fips=true** です。

注記

JVM インスタンスの実行中に FIPS モードがシステムで有効になっている場合は、変更を有効にするためにインスタンスを再起動する必要があります。

FIPS モードを有効にする方法は、FIPS モードへのシステムの切り替え を参照してください。

Red Hat build of OpenJDK 17 を設定して、グローバル FIPS 調整をバイパスできます。たとえば、Red Hat build of OpenJDK が提供するスキームではなく、Hardware Security Module (HSM) で FIPS コンプライアンスを有効にする場合があります。

Red Hat build of OpenJDK 17 の FIPS プロパティーは次のとおりです。

- **security.useSystemPropertiesFile**
  - セキュリティープロパティーは、**$JAVA_HOME/conf/security/java.security** または **java.security.properties** で指定したファイルにあります。
  - デフォルトの **java.security** ファイル内の値を変更するには、特権アクセスが必要です。
  - 永続設定です。
  - **false** に設定すると、グローバル FIPS と crypto-policies 調整の両方が無効になります。デフォルトでは **true** に設定されます。
- **java.security.disableSystemPropertiesFile**
  - JVM に渡されるシステムプロパティーを引数として渡すシステムプロパティー。たとえば、**-Djava.security.disableSystemPropertiesFile=true** です。
  - 非特権アクセスで十分です。
  - 非永続的な設定です。
  - **true** に設定すると、グローバル FIPS と crypto-policies アライメントの両方が無効になります。**security.useSystemPropertiesFile=false** セキュリティープロパティーと同じ効果が生成されます。いずれのプロパティーも異なる動作に設定されている場合は、**java.security.disableSystemPropertiesFile** が上書きされます。デフォルトでは **false** に設定されます。
- **com.redhat.fips**
  - JVM に渡されるシステムプロパティーを引数として渡すシステムプロパティー。たとえば、**-Dcom.redhat.fips=false** です。
  - 非特権アクセスで十分です。
  - 非永続的な設定です。
  - **false** に設定すると、グローバル crypto-policies を適用している間に FIPS 調整を無効にし

ます。以前のプロパティーのいずれかが crypto-policies 調整を無効にするように設定されていると、このプロパティーは効果がありません。つまり、crypto-policies は FIPS 調整の前提条件です。デフォルトでは **true** に設定されます。

第3章 RED HAT ビルドの OPENJDK 11 のデフォルトの FIPS 設定

セキュリティープロバイダー

Red Hat ビルドの OpenJDK セキュリティーポリシーは、グローバル java セキュリティーポリシーファイルによって制御されます。**$JRE_HOME/lib/security/java.security** に java セキュリティーポリシーファイルがあります。

FIPS モードを有効にすると、Red Hat build of OpenJDK はインストールされたセキュリティープロバイダーを以下のものに置き換えます。これは優先度が高い順になります。

SunPKCS11-NSS-FIPS

- Network Security Services (NSS) ソフトウェアトークン (PKCS#11 バックエンド) で初期化されます。NSS Software Token が以下のように設定されます。
  - name = NSS-FIPS
  - nssLibraryDirectory = /usr/lib64
  - nssSecmodDirectory = /etc/pki/nssdb
  - nssDbMode = readOnly
  - nssModule = fips
- NSS ライブラリーは、FIPS 準拠のソフトウェアトークンを実装します。また、RHEL で FIPS ポリシー対応も可能にします。

SUN

- X.509 証明書でサポートされるのは、X.509 証明書のみです。アプリケーションがこのプロバイダーの他の暗号化アルゴリズムを使用していないことを確認します。たとえば、**MessageDigest.getInstance("SHA-256", Security.getProvider("SUN"))** は機能しますが、FIPS に準拠していない **MessageDigest** サービスにつながることになります。

SunEC

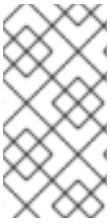
- SunPKCS11 補助ヘルパーの場合のみ。アプリケーションがこのプロバイダーを明示的に使用していないことを確認してください。

SunJSSE

- キー派生など、TLS エンジンが必要とするすべての暗号化プリミティブに対して **SunPKCS11-NSS-FIPS** プロバイダーで初期化されます。

crypto-policies

FIPS モードを有効にすると、Red Hat build of OpenJDK はグローバル crypto-policies から暗号化アルゴリズムの設定値を取ります。これらの値は **/etc/crypto-policies/back-ends/java.config** にあります。RHEL の **update-crypto-policies** ツールを使用すると、一貫性のある方法で crypto-policies を管理できます。



注記

暗号ポリシー承認アルゴリズムは、Red Hat build of OpenJDK の FIPS モードでは使用できません。これは、FIPS 準拠の実装が NSS ライブラリーで利用できない場合や、Red Hat build of OpenJDK の SunPKCS11 セキュリティープロバイダーでサポートされない場合に発生します。

trust Anchor 証明書

Red Hat build of OpenJDK は、FIPS モードでは、グローバル Trust Anchor 証明書リポジトリーを使用します。このリポジトリーは、**/etc/pki/java/cacerts** で確認することができます。RHEL の **update-ca-trust** ツールを使用して、一貫性のある方法で証明書を管理します。

キーストア

FIPS モードでは、Red Hat build of OpenJDK は、鍵の読み取り専用 **PKCS#11** ストアとして NSS DB を使用します。これにより、**keystore.type** セキュリティープロパティーは **PKCS11** に設定されます。NSS DB リポジトリーは **/etc/pki/nssdb** で見つけることができます。RHEL で **modutil** ツールを使用し、NSS DB キーを管理します。

SunPKCS11 プロバイダー設定属性

SunPKCS11 プロバイダーには、キーオブジェクトなどのネイティブリソースの使用を強化する設定属性が含まれています。**SunPKCS11** プロバイダーは、ネイティブの **PKCS11** ライブラリーと機能するためにネイティブリソースを使用する必要があります。

表3.1 SunPKCS11 プロバイダー設定属性

属性	タイプ	説明
destroyTokenAfterLogout	ブール値	デフォルトは **false** です。true に設定すると、アプリケーションが SunPKCS11 プロバイダーインスタンスの **logout ()** メソッドを呼び出すと、基礎となるトークンオブジェクトが SunPKCS11 プロバイダーインスタンスによって削除され、リソースが解放されます。これにより、**logout ()** メソッド呼び出しの実行後に **SunPKCS11** プロバイダーが使用できなくなるため、PKCS11 をシステムプロバイダーリストに追加しないでください。
cleaner.shortInterval	integer	デフォルトは **2000** ミリ秒 (ms) です。この属性は、クリーナースレッドが不要になったネイティブ PKCS11 参照をクリアキューから削除してネイティブメモリーを解放する頻度を定義します。 **注:** クリアキューにネイティブ PKCS11 参照が存在せず、クリーナースレッドがキューで削除プロセスを 200 回以上試行する場合、クリーナースレッドは、**cleaner.longInterval** 属性値を使用するように切り替わります。
cleaner.longInterval	integer	デフォルトは **60000** ミリ秒 (ms) です。この属性は、ビジーではない期間中に、クリーナースレッドがネイティブ PKCS11 参照のクリアキューをチェックする頻度を定義します。 **注記:** スレッドがクリアキューのネイティブ PKCS11 参照を検出すると、クリーナースレッドは **cleaner.shortInterval** 属性値を使用します。

改訂日時: 2024-05-10