



Red Hat build of OpenJDK 17

Configuring Red Hat build of OpenJDK 17 on RHEL with FIPS

Red Hat build of OpenJDK 17 Configuring Red Hat build of OpenJDK 17 on RHEL with FIPS

Legal Notice

Copyright © 2024 Red Hat, Inc.

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.

Abstract

Red Hat build of OpenJDK is a Red Hat offering on the Red Hat Enterprise Linux platform. The Configuring Red Hat build of OpenJDK 17 on RHEL with FIPS guide provides an overview of FIPS and explains how to enable and configure Red Hat build of OpenJDK with FIPS.

Table of Contents

PROVIDING FEEDBACK ON RED HAT BUILD OF OPENJDK DOCUMENTATION	3
MAKING OPEN SOURCE MORE INCLUSIVE	4
CHAPTER 1. INTRODUCTION TO FEDERAL INFORMATION PROCESSING STANDARDS (FIPS)	5
CHAPTER 2. FIPS SETTINGS IN RED HAT BUILD OF OPENJDK 17	6
CHAPTER 3. FIPS AUTOMATION IN RED HAT BUILD OF OPENJDK 17	10
3.1. SECURITY PROVIDERS	10
3.2. CRYPTO-POLICIES	11
3.3. TRUST ANCHOR CERTIFICATES	11
3.4. KEYSTORES	11

PROVIDING FEEDBACK ON RED HAT BUILD OF OPENJDK DOCUMENTATION

To report an error or to improve our documentation, log in to your Red Hat Jira account and submit an issue. If you do not have a Red Hat Jira account, then you will be prompted to create an account.

Procedure

1. Click the following link to [create a ticket](#).
2. Enter a brief description of the issue in the **Summary**.
3. Provide a detailed description of the issue or enhancement in the **Description**. Include a URL to where the issue occurs in the documentation.
4. Clicking **Submit** creates and routes the issue to the appropriate documentation team.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

CHAPTER 1. INTRODUCTION TO FEDERAL INFORMATION PROCESSING STANDARDS (FIPS)

The Federal Information Processing Standards (FIPS) provides guidelines and requirements for improving security and interoperability across computer systems and networks. The FIPS 140-2 and 140-3 series apply to cryptographic modules at both the hardware and software levels. The National Institute of Standards and Technology in the United States implements a cryptographic module validation program with searchable lists of both in-process and approved cryptographic modules.

Red Hat Enterprise Linux (RHEL) brings an integrated framework to enable FIPS 140-2 compliance system-wide. When operating under FIPS mode, software packages using cryptographic libraries are self-configured according to the global policy. Most of the packages provide a way to change the default alignment behavior for compatibility or other needs.

Red Hat build of OpenJDK 17 is a FIPS policy-aware package.

Additional resources

- For more information about the cryptographic module validation program, see [Cryptographic Module Validation Program CMVP](#) on the *National Institute of Standards and Technology* website.
- For more information on how to install RHEL with FIPS mode enabled, see [Installing a RHEL 8 system with FIPS mode enabled](#).
- For more information on how to enable FIPS mode after installing RHEL, see [Switching the system to FIPS mode](#).
- For more information on how to run Red Hat build of OpenJDK in FIPS mode on RHEL. See [Running OpenJDK in FIPS mode on RHEL](#).
- For more information on Red Hat compliance with Government Standards, see [Government Standards](#).

CHAPTER 2. FIPS SETTINGS IN RED HAT BUILD OF OPENJDK 17

At startup, Red Hat build of OpenJDK 17 checks if the system FIPS policy is enabled. If this policy is enabled, Red Hat build of OpenJDK 17 performs a series of automatic configurations that are intended to help Java applications to comply with FIPS requirements.

These automatic configurations include the following actions:

- Installing a restricted list of security providers that contains the FIPS-certified Network Security Services (NSS) software token module for cryptographic operations
- Enforcing the Red Hat Enterprise Linux (RHEL) FIPS crypto-policy for Java that limits the algorithms and parameters available



NOTE

If FIPS mode is enabled in the system while a JVM instance is running, the JVM instance must be restarted to allow changes to take effect.

You can configure Red Hat build of OpenJDK 17 to bypass the described FIPS automation. For example, you might want to achieve FIPS compliance through a Hardware Security Module (HSM) instead of the NSS software token module.

You can specify FIPS configurations by using *system* or *security* properties.

To better understand FIPS properties, you must understand the following JDK property classes:

- System properties are JVM arguments prefixed with **-D**, which generally take the form of **-Dproperty.name=property.value**. Privileged access is not required to pass any of these values. Only the launched JVM is affected by the configuration, and persistence depends on the existence of a launcher script. UTF-8 encoded values are valid for system properties.
- Security properties are available in **\$JRE_HOME/conf/security/java.security** or in the file that the **java.security.properties** system property points to. Privileged access is required to modify values in the **\$JRE_HOME/conf/security/java.security** file. Any modification to this file persists and affects all instances of the same Red Hat build of OpenJDK 17 deployment. Non-Basic Latin Unicode characters must be encoded with **\uXXXX**.

When system and security properties have the same name and are set to different values, the system property takes precedence. Depending on their configuration, properties might affect other properties with different names.

For more information about security properties and their default values, see the **java.security** file.

The following list details properties that affect the FIPS configuration for Red Hat build of OpenJDK 17:

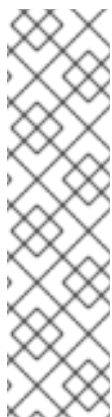
Property	Type	Default value	Description
security.useSystemPropertiesFile	Security	true	When set to false , this property disables the FIPS automation, which includes global crypto-policies alignment.

Property	Type	Default value	Description
java.security.disableSystemPropertiesFile	System	false	When set to true , this property disables the FIPS automation, which includes global crypto-policies alignment. This has the same effect as a security.useSystemPropertiesFile=false security property. If both properties are set to different behaviors, java.security.disableSystemPropertiesFile takes precedence.
com.redhat.fips	System	true	When set to false , this property disables the FIPS automation while still enforcing the FIPS crypto-policy. If any of the preceding properties are set to disable the FIPS automation, this property has no effect. Crypto-policies are a prerequisite for FIPS automation.
fips.keystore.type	Security	PKCS12	This property sets the default keystore type when Red Hat build of OpenJDK 17 is in FIPS mode. Supported values are PKCS12 and PKCS11 .

In addition to the previously described settings, specific configurations can be applied to use NSS DB keystores in FIPS mode. These keystores are handled by the **SunPKCS11** security provider and the NSS software token, which is the security provider's **PKCS#11** back end.

The following list details the NSS DB FIPS properties for Red Hat build of OpenJDK 17:

Property	Type	Default value	Description
fips.nssdb.path	System or Security	sql:/etc/pki/nssdb	File-system path that points to the NSS DB location. The syntax for this property is identical to the nssSecmodDirectory attribute available in the SunPKCS11 NSS configuration file. The property allows an sql: prefix to indicate that the referred NSS DB is of SQLite type.

Property	Type	Default value	Description
fips.nssdb.pin	System or Security	pin: (empty PIN)	<p>PIN (password) for the NSS DB that fips.nssdb.path points to.</p> <p>You can use this property to pass the NSS DB PIN in one of the following forms:</p> <ul style="list-style-type: none"> ● pin:<value> In this situation, <value> is a clear text PIN value (for example, pin:1234abc). ● env:<value> In this situation, <value> is an environment variable that contains the PIN value (for example, env:NSSDB_PIN_VAR). ● file:<value> In this situation, <value> is the path to a UTF-8 encoded file that contains the PIN value in its first line (for example, file:/path/to/pin.txt). <p>The pin:<value> option accommodates both cases in which the PIN value is passed as a JVM argument or programmatically through a system property. Programmatic setting of the PIN value provides flexibility for applications to decide how to obtain the PIN.</p> <p>The file:<value> option is compatible with NSS modutil -pwfile and -newpwfile arguments, which are used for an NSS DB PIN change.</p> <div data-bbox="906 1473 1018 1888" style="border: 1px solid black; padding: 5px; width: fit-content;">  </div> <p>NOTE</p> <p>If a cryptographic operation requires NSS DB authentication and the status is not authenticated, Red Hat build of OpenJDK 17 performs an implicit login with this PIN value. An application can perform an explicit login by invoking KeyStore::load before any cryptographic operation.</p>



IMPORTANT

Perform a security assessment, so that you can decide on a configuration that protects the integrity and confidentiality of the stored keys and certificates. This assessment should consider threats, contextual information, and other security measures in place, such as operating system user isolation and file-system permissions. For example, default configuration values might not be appropriate for an application storing keys and running in a multi-user environment. Use the **modutil** tool in RHEL to create and manage NSS DB keystores, and use **certutil** or **keytool** to import certificates and keys.

Additional resources

- For more information about enabling FIPS mode, see [Switching the system to FIPS mode](#) .

CHAPTER 3. FIPS AUTOMATION IN RED HAT BUILD OF OPENJDK 17

This chapter describes how the FIPS automation is implemented in Red Hat build of OpenJDK 17 and how FIPS automation might impact your applications.

3.1. SECURITY PROVIDERS

When FIPS mode is enabled, Red Hat build of OpenJDK 17 replaces the installed security providers with a constrained list. Some security services and algorithms might be dropped, so that only a FIPS-certified module performs cryptographic operations. The following list describes installed security providers, services, algorithms and enabled configurations:

SunPKCS11-NSS-FIPS

Initialized with the NSS software token, which is the service provider's **PKCS#11** back end, in accordance with the configuration found at `$JRE_HOME/conf/security/nss.fips.cfg`:

- **name = NSS-FIPS**
- **nssLibraryDirectory = /usr/lib64**
- **nssSecmodDirectory = \${fips.nssdb.path}**
- **nssDbMode = readWrite**
- **nssModule = fips**
- **attributes(*,CKO_SECRET_KEY,CKK_GENERIC_SECRET)={ CKA_SIGN=true }**



NOTE

Changes to this configuration are discouraged.

All cryptographic services are enabled. These include **AlgorithmParameters**, **Cipher**, **KeyAgreement**, **KeyFactory**, **KeyGenerator**, **KeyPairGenerator**, **KeyStore**, **Mac**, **MessageDigest**, **SecretKeyFactory**, **SecureRandom**, and **Signature**.

SUN

Only X.509 certificate-related (**CertificateFactory**, **CertPathBuilder**, **CertPathValidator**, **CertStore**), **AlgorithmParameterGenerator**, **AlgorithmParameters**, and **KeyStore** (**JKS**, **PKCS12**) services are enabled.

SunEC

Only **AlgorithmParameters** and **KeyFactory** services are enabled.

SunJSSE

Only TLS-related services (**KeyManagerFactory**, **SSLContext**, **TrustManagerFactory**) and **KeyStore** (**PKCS12**) are enabled.

SunJCE

Only **AlgorithmParameterGenerator**, **AlgorithmParameters**, **KeyFactory**, and **SecretKeyFactory** (except **BKDF2** algorithms) services are enabled.

SunRsaSign

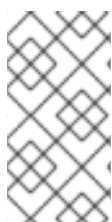
Only **AlgorithmParameters** and **KeyFactory** services are enabled.

XMLDSig

All services are enabled. These include **TransformService**, **KeyInfoFactory**, and **XMLSignatureFactory**.

3.2. CRYPTO-POLICIES

In FIPS mode, Red Hat build of OpenJDK 17 takes the list of disabled cryptographic algorithms and other configurations from the global FIPS crypto-policy in RHEL. You can find these values at **/etc/crypto-policies/back-ends/java.config**. You can use the **update-crypto-policies** tool from RHEL to consistently manage crypto-policies.



NOTE

A crypto-policies approved algorithm might not be usable when Red Hat build of OpenJDK is configured in FIPS mode. This occurs when a FIPS-certified implementation is not available in the NSS software token or when it is not supported in the **SunPKCS11** security provider.

3.3. TRUST ANCHOR CERTIFICATES

In FIPS mode, Red Hat build of OpenJDK 17 uses the global Trust Anchor certificates repository by default. This behavior is equivalent to non-FIPS mode. This repository is located at **/etc/pki/java/cacerts**. Use the **update-ca-trust** tool from RHEL to consistently manage certificates. Optionally, you can store Trust Anchor certificates in your own **PKCS12** and **PKCS11** keystores, and use them for TLS communication. For more information, see the [TrustManagerFactory::init](#) documentation.

When the **javax.net.ssl.trustStoreType** system property is not set and FIPS mode is enabled, Red Hat build of OpenJDK 17 automatically sets this system property to the value of the **keystore.type** security property. This behavior is equivalent to non-FIPS mode.

3.4. KEYSTORES

In FIPS mode, Red Hat build of OpenJDK 17 enables the use of the **PKCS12** and **PKCS11** keystore types. **PKCS12** is used by default. You can change the default keystore type by using the **fips.keystore.type** security property. An application can also select which keystore type to use when invoking **KeyStore.getInstance(<type>)**.

When opening a **PKCS11** keystore, Red Hat build of OpenJDK 17 uses the SQLite NSS DB located at **/etc/pki/nssdb**. This NSS DB might be unsuitable to store keys. You can specify a different database by setting a value for the **fips.nssdb.path** property. For more information and security considerations, see [FIPS settings in Red Hat build of OpenJDK 17](#).

When you set the **fips.keystore.type** security property to **PKCS11** and FIPS mode is enabled, Red Hat build of OpenJDK 17 automatically assigns the **javax.net.ssl.keyStore** system property to a value of **NONE**. This behavior facilitates the use of **PKCS#11** keystores by saving a manual configuration step. For more information, see [JDK-8238264](#).

Revised on 2024-05-28 07:35:02 UTC