Red Hat build of OpenJDK 11

Configuring Red Hat build of OpenJDK 11 on RHEL with FIPS

Last Updated: 2024-05-09

Legal Notice

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 11 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

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 11 is a FIPS policy-aware package.

Additional resources

- 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. CONFIGURE RED HAT BUILD OF OPENJDK 11 IN FIPS MODE

Red Hat build of OpenJDK 11 checks if the FIPS mode is enabled in the system at startup. If yes, it self-configures FIPS according to the global policy. This is the default behavior since RHEL 8.3. Previous RHEL 8 releases require the **com.redhat.fips** system property set to **true** as a JVM argument. For example, **-Dcom.redhat.fips=true**.

NOTE

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

For more information on how to enable FIPS mode, see Switching the system to FIPS mode .

You can configure Red Hat build of OpenJDK 11 to bypass the global FIPS alignment. For example, you might want to enable FIPS compliance through a Hardware Security Module (HSM) instead of the scheme provided by Red Hat build of OpenJDK.

Following are the FIPS properties for Red Hat build of OpenJDK 11:

- **security.useSystemPropertiesFile**
  - Security property located at **$JAVA_HOME/conf/security/java.security** or in the file directed to **java.security.properties**.
  - Privileged access is required to modify the value in the default **java.security** file.
  - Persistent configuration.
  - When set to **false**, both the global FIPS and the crypto-policies alignment are disabled. By default, it is set to **true**.
- **java.security.disableSystemPropertiesFile**
  - System property passed to the JVM as an argument. For example, **-Djava.security.disableSystemPropertiesFile=true**.
  - Non-privileged access is enough.
  - Non-persistent configuration.
  - When set to **true**, both the global FIPS and the crypto-policies alignment are disabled; generating the same effect than a **security.useSystemPropertiesFile=false** security property. If both properties are set to different behaviors, **java.security.disableSystemPropertiesFile** overrides. By default, it is set to **false**.
- **com.redhat.fips**
  - System property passed to a JVM as an argument. For example, **-Dcom.redhat.fips=false**.
  - Non-privileged access is enough.
  - Non-persistent configuration.
  - When set to **false**, disables the FIPS alignment while still applying the global crypto-policies.

If any of the previous properties is set to disable the crypto-policies alignment, this property has no effect. In other words, crypto-policies is a prerequisite for FIPS alignment. By default, it is set to **true**.

CHAPTER 3. DEFAULT FIPS CONFIGURATIONS IN RED HAT BUILD OF OPENJDK 11

Security providers

The Red Hat build of OpenJDK security policy is controled by the global java security policy file. You can find the java security policy file at **$JRE_HOME/lib/security/java.security**.

With FIPS mode enabled, Red Hat build of OpenJDK replaces the installed security providers with the following ones (in descending priority order):

SunPKCS11-NSS-FIPS

- Initialized with a Network Security Services (NSS) Software Token (PKCS#11 backend). The NSS Software Token is configured as follows:
  - name = NSS-FIPS
  - nssLibraryDirectory = /usr/lib64
  - nssSecmodDirectory = /etc/pki/nssdb
  - nssDbMode = readOnly
  - nssModule = fips
- The NSS library implements a FIPS-compliant Software Token. Also, FIPS policy-aware in RHEL.

SUN

- For X.509 certificates support only. Make sure that your application is not using other cryptographic algorithms from this provider. For example, **MessageDigest.getInstance("SHA-256", Security.getProvider("SUN"))** would work but lead to a non-FIPS compliant **MessageDigest** service.

SunEC

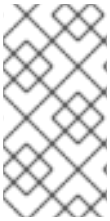
- For SunPKCS11 auxiliary helpers only. Make sure that your application is not explicitly using this provider.

SunJSSE

- Initialized with the **SunPKCS11-NSS-FIPS** provider for all cryptographic primitives required by the TLS engine, including key derivation.

Crypto-policies

With FIPS mode enabled, Red Hat build of OpenJDK takes configuration values of cryptographic algorithms from global crypto-policies. You can find these values at **/etc/crypto-policies/back-ends/java.config**. You can use the **update-crypto-policies** tooling from RHEL to manage crypto-policies in a consistent way.



NOTE

A crypto-policies approved algorithm might not be usable in Red Hat build of OpenJDK's FIPS mode. This occurs when a FIPS-compliant implementation is not available in the NSS library or when it is not supported in Red Hat build of OpenJDK's SunPKCS11 security provider.

Trust Anchor certificates

Red Hat build of OpenJDK uses the global Trust Anchor certificates repository when in FIPS mode. You can locate this repository at **/etc/pki/java/cacerts**. Use the **update-ca-trust** tooling from RHEL to manage certificates in a consistent way.

Key store

With FIPS mode, Red Hat build of OpenJDK uses the NSS DB as a read-only **PKCS#11** store for keys. As a result, the **keystore.type** security property is set to **PKCS11**. You can locate the NSS DB repository at **/etc/pki/nssdb**. Use the **modutil** tooling in RHEL to manage NSS DB keys.

SunPKCS11 provider configuration attributes

The **SunPKCS11** provider includes configuration attributes that enhance the usage of native resources, such as key objects. The **SunPKCS11** provider must use its native resources to work with native **PKCS11** libraries.

Table 3.1. SunPKCS11 provider configuration attributes

Attribute	Type	Description
destroyTokenAfterLogout	boolean	Defaults to **false**. If set to true, when an application invokes the **logout()** method of the SunPKCS11 provider instance, the underlying token object is deleted by the SunPKCS11 provider instance and resources are released. This renders the **SunPKCS11** provider unusable after execution of **logout()** method calls, so do not add the PKCS11 to the system provider list.
cleaner.shortInterval	integer	Defaults to **2000** milliseconds (ms). The attribute defines the frequency that a cleaner thread removes no-longer-needed native PKCS11 references from the clearing queue to free native memory. **Note:** The cleaner thread switches to use the **cleaner.longInterval** attribute value if no native PKCS11 references exist in the clearing queue and the cleaner thread attempts the removal process on the queue more than 200 times.
cleaner.longInterval	integer	Defaults to **60000** milliseconds (ms). The attribute defines the frequency that a cleaner thread checks the clearing queue for native PKCS11 references during non-busy periods of time. **Note:** The cleaner thread switches to use the **cleaner.shortInterval** attribute value when the thread detects native PKCS11 references in the clearing queue.

Revised on 2024-05-09 16:46:03 UTC