Oracle® Jipher

Release Notes

Release 20

G59618-01

September 2026

Oracle Jipher 20 Release Notes

Oracle Jipher 20

Oracle Jipher is a Java Cryptographic Service Provider (CSP) that dynamically loads a Federal Information Processing Standards (FIPS) 140-3 validated OpenSSL cryptographic module at runtime and uses it as the underlying cryptographic implementation. It enables the deployment of Java applications in FIPS-regulated environments. Jipher makes its cryptographic services available to Java developers using the Java Cryptography Architecture (JCA).

Oracle Jipher is offered as part of the Oracle Java SE Universal Subscription and to Oracle customers running Java workloads in Oracle Cloud Infrastructure (OCI). Note that the Jipher feature included in the Java SE Subscription has a shorter support lifecycle than the corresponding supported JDK feature versions. Please refer to the Java SE Support Roadmap for more information.

Supported Runtimes

- Oracle JDK 27
- Oracle JDK 25
- Oracle JDK 21
- Oracle JDK 17

Supported Platforms

- Oracle Linux 10, 9, and 8 on x86-64 and aarch64
- Red Hat Enterprise Linux (RHEL) 10, 9, and 8 on x86-64 and aarch64
- macOS 26 on M series Apple silicon
- Windows Server 2025

Important Changes and Information

- **Runtime Native Library Dependencies**

  In Jipher 10.*x*, Jipher's runtime native library dependencies are embedded in the `jipher-jce` JAR. At runtime, the native libraries appropriate for the platform on which Jipher is deployed are automatically extracted from the `jipher-jce` JAR file to a temporary directory in the file system and loaded from there into the JVM process.

  In Jipher 20, Jipher's runtime native library dependencies are not embedded in the `jipher-jce` JAR. They are distributed in a separate `jipher-native` package. The native library dependencies appropriate for the platform on which Jipher is deployed must be unpacked from the `jipher-native` package to the file system before the `jipher-jce` JAR is loaded by a JVM.

  See Jipher Deployment for more information.

- **Java Module System**

  The library is now delivered as an explicit module rather than an automatic module. The `com.oracle.jipher` module does not export any packages.

  Applications that list the `jipher-jce` JAR on the module path can no longer create an instance of the `com.oracle.jipher.provider.JipherJCE` class directly or through reflection:

  ```
  import com.oracle.jipher.provider.JipherJCE;
  ...
  // Cannot create an instance of JipherJCE directly
  Provider provider = new com.oracle.jipher.provider.JipherJCE();

  // Cannot create an instance of JipherJCE through reflection
  Class<?> jipherClass =
  Class.forName("com.oracle.jipher.provider.JipherJCE");
  Provider provider = (Provider)
  jipherClass.getDeclaredConstructor().newInstance();
  ```

  An instance of the JipherJCE provider can instead be created using the service loader:

  ```
  Provider provider = ServiceLoader.load(Provider.class).stream()
      .filter(provider ->
  provider.type().getName().equals("com.oracle.jipher.provider.JipherJ
  CE"))
      .map(ServiceLoader.Provider::get)
      .findFirst()
      .orElseThrow(() -> new IllegalStateException("JipherJCE
  provider not found"));
  ```

- **The Jipher Provider is Statically Registered by its Provider Name, not its Class Name**

Applications that list the `jipher-jce` JAR on the module path can no longer statically register the JipherJCE provider using its class name in the security properties file (which is, by default `$JAVA_HOME/conf/security/java.security`; see The Security Properties File in *Java Platform, Standard Edition Security Developer's Guide*):

```
# Invalid: Cannot register the JipherJCE provider using its class
name
security.provider.1=com.oracle.jipher.provider.JipherJCE
```

Applications should statically register the JipherJCE provider using its provider name, for example:

```
security.provider.1=JipherJCE
```

See Registering Jipher as the Most Preferred Provider for more information.

- **Support for Static Methods `isAvailable` and `loadingException`**
  The static methods `JipherJCE.isAvailable()` and `JipherJCE.loadingException()`, which were present in Jipher 10.*x*, have been removed in Jipher 20.

  In Jipher 10.*x*, an application could use these methods to determine whether OpenSSL was available and, if not, why OpenSSL could not be loaded.

  In Jipher 20, an application can determine whether OpenSSL is available by attempting to load `JipherJCE` with `ServiceLoader`. If the load succeeds, then OpenSSL is available. If it throws a `ServiceConfigurationError`, then the reason OpenSSL could not be loaded is available from the final `Throwable` in the chain of exceptions:

```
try {
    ServiceLoader.load(Provider.class).stream()
        .filter(provider ->
provider.type().getName().equals("com.oracle.jipher.provider.JipherJ
CE"))
        .map(ServiceLoader.Provider::get)
        .findFirst()
        .orElseThrow(() -> new IllegalStateException("JipherJCE
provider not found"));
} catch (ServiceConfigurationError sce) {
    if (sce.getCause() instanceof ProviderException
providerException) {
        providerException.printStackTrace();
    }
}
```

  See Reporting Misconfiguration for more information.

- **Addition of Support for Post-Quantum Cryptographic (PQC) Algorithms**

  Jipher 20 adds support for Module-Lattice (ML) based algorithms:

  - Key Encapsulation Mechanisms (KEM): ML-KEM, ML-KEM-512, ML-KEM-1024
  - Digital signature algorithms (DSA): ML-DSA-44, ML-DSA-65, ML-DSA-87

  `KeyFactory`, `KeyPairGenerator`, `KEM`, and `Signature` service support is provided.

  See Supported Algorithm Strings for more information.

- **Addition of Support for Key Derivation Function (KDF) Service Algorithms**

  Jipher 20 adds support for HMAC-based Key Derivation Function (HKDF) KDF service algorithms.

  See Supported Algorithm Strings for more information.

- **Removal of Support for HmacPBE* Mac Algorithms**

  By default, Jipher 20 does not provide HmacPBE* Mac algorithms (see Mac Algorithms in *Java Security Standard Algorithm Names*), as defined in Appendix B.4: Keys for Password Integrity Mode in RFC 7292. These password-based MAC algorithms are not allowed by FIPS 140-3.

  On JDK 25 and earlier, the SUN provider's PKCS12 `KeyStore` service employs HmacPBE* Mac algorithms for the integrity check when reading `KeyStore` instances from or writing them to persistent storage. If your application absolutely must load a PKCS12 `KeyStore` from or store one to persistent storage, then you can set the system property `jipher.nonfips.supportPkcs12Kdf` to `true` to configure Jipher to provide support for HmacPBE* Mac algorithms. This setting deviates from the FIPS 140-3 standard.

  See Optionally Supported Non-FIPS 140-3 Allowed Algorithms and Configuring Jipher Through System and Security Properties for more information.

- **Removal of Support for the PBEWithSHA1AndDESede Algorithm**

  By default, Jipher 20 does not provide the PBEWithSHA1AndDESede algorithm. This password-based algorithm is not allowed by FIPS 140-3. Jipher can be configured to provide support for the PBEWithSHA1AndDESede algorithm by setting the system property `jipher.nonfips.supportPkcs12Kdf` to `true`.

  See Optionally Supported Non-FIPS 140-3 Allowed Algorithms and Configuring Jipher Through System and Security Properties for more information.

Additional Information

- Oracle Jipher Documentation
- Download Jipher from Java Verified Portfolio
- For support, refer to My Oracle Support

Oracle Jipher Release Notes, Release 20
G59618-01

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.