**How to Make Applications "Exempt" from Cryptographic Restrictions**

**Note**: This section should be ignored by most application developers. It is only for people whose applications may be exported to those few countries whose governments mandate cryptographic restrictions, if it is desired that such applications have fewer cryptographic restrictions than those mandated.

By default, an application can use cryptographic algorithms of any strength. However, due to import control restrictions by the governments of a few countries, you may have to limit those algorithms' strength. The JCA framework includes an ability to enforce restrictions regarding the maximum strengths of cryptographic algorithms available to applications in different jurisdiction contexts (locations). You specify these restrictions in jurisdiction policy files. For more information about jurisdiction policy files and how to create and configure them, see [Appendix B: Jurisdiction Policy File Format](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#AppB) and [Appendix C: Cryptographic Strength Configuration](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#AppC).

It is possible that the governments of some or all such countries may allow certain applications to become exempt from some or all cryptographic restrictions. For example, they may consider certain types of applications as "special" and thus exempt. Or they may exempt any application that utilizes an "exemption mechanism," such as key recovery. Applications deemed to be exempt could get access to stronger cryptography than that allowed for non-exempt applications in such countries.

For an application to be recognized as "exempt" at runtime, it must meet the following conditions:

* It must have a permission policy file bundled with it in a JAR file. The permission policy file specifies what cryptography-related permissions the application has, and under what conditions (if any).
* The JAR file containing the application and the permission policy file must have been signed using a code-signing certificate issued after the application was accepted as exempt.

Below are sample steps required in order to make an application exempt from some cryptographic restrictions. This is a basic outline that includes information about what is required by JCA in order to recognize and treat applications as being exempt. You will need to know the exemption requirements of the particular country or countries in which you would like your application to be able to be run but whose governments require cryptographic restrictions. You will also need to know the requirements of a JCA framework vendor that has a process in place for handling exempt applications. Consult such a vendor for further information.

**Note**: The SunJCE provider does not supply an implementation of the ExemptionMechanismSpi class.

* Step 1: Write and Compile Your Application Code
* Step 2: Create a Permission Policy File Granting Appropriate Cryptographic Permissions
* Step 3: Prepare for Testing
  + Step 3a: Apply for Government Approval From the Government Mandating Restrictions.
  + Step 3b: Get a Code-Signing Certificate
  + Step 3c: Bundle the Application and Permission Policy File into a JAR file
  + Step 3d: [Sign the JAR file](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/HowToImplAProvider.html#Step61)
  + Step 3e: Set Up Your Environment Like That of a User in a Restricted Country
  + Step 3f: (only for applications using exemption mechanisms) Install a Provider Implementing the Exemption Mechanism Specified by the Entry in the Permission Policy File
* Step 4: Test Your Application
* Step 5: Apply for U.S. Government Export Approval If Required
* Step 6: Deploy Your Application

**Special Code Requirements for Applications that Use Exemption Mechanisms**

When an application has a permission policy file associated with it (in the same JAR file) and that permission policy file specifies an exemption mechanism, then when the Cipher getInstance method is called to instantiate a Cipher, the JCA code searches the installed providers for one that implements the specified exemption mechanism. If it finds such a provider, JCA instantiates an ExemptionMechanism object associated with the provider's implementation, and then associates the ExemptionMechanism object with the Cipher returned by getInstance.

After instantiating a Cipher, and prior to initializing it (via a call to the Cipher init method), your code must call the following Cipher method:

public ExemptionMechanism getExemptionMechanism()

This call returns the ExemptionMechanism object associated with the Cipher. You must then initialize the exemption mechanism implementation by calling the following method on the returned ExemptionMechanism:

public final void init(Key key)

The argument you supply should be the same as the argument of the same types that you will subsequently supply to a Cipher init method.

Once you have initialized the ExemptionMechanism, you can proceed as usual to initialize and use the Cipher.

**Permission Policy Files**

In order for an application to be recognized at runtime as being "exempt" from some or all cryptographic restrictions, it must have a permission policy file bundled with it in a JAR file. The permission policy file specifies what cryptography-related permissions the application has, and under what conditions (if any).

**NOTE:** The permission policy file bundled with an application must be named **cryptoPerms**.

The format of a permission entry in a permission policy file that accompanies an exempt application is the same as the format for a jurisdiction policy file downloaded with the JDK, which is:

permission <crypto permission class name>[ <alg\_name>

[[, <exemption mechanism name>][, <maxKeySize>

[, <AlgorithmParameterSpec class name>,

<parameters for constructing an AlgorithmParameterSpec object>

]]]];

See [Appendix B](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#AppB) for more information about the jurisdiction policy file format.

**Permission Policy Files for Exempt Applications**

Some applications may be allowed to be completely unrestricted. Thus, the permission policy file that accompanies such an application usually just needs to contain the following:

grant {

// There are no restrictions to any algorithms.

permission javax.crypto.CryptoAllPermission;

};

If an application just uses a single algorithm (or several specific algorithms), then the permission policy file could simply mention that algorithm (or algorithms) explicitly, rather than granting CryptoAllPermission. For example, if an application just uses the Blowfish algorithm, the permission policy file doesn't have to grant CryptoAllPermission to all algorithms. It could just specify that there is no cryptographic restriction if the Blowfish algorithm is used. In order to do this, the permission policy file would look like the following:

grant {

permission javax.crypto.CryptoPermission "Blowfish";

};

**Permission Policy Files for Applications Exempt Due to Exemption Mechanisms**

If an application is considered "exempt" if an exemption mechanism is enforced, then the permission policy file that accompanies the application must specify one or more exemption mechanisms. At runtime, the application will be considered exempt if any of those exemption mechanisms is enforced. Each exemption mechanism must be specified in a permission entry that looks like the following:

// No algorithm restrictions if specified

// exemption mechanism is enforced.

permission javax.crypto.CryptoPermission \*,

"<ExemptionMechanismName>";

where <ExemptionMechanismName> specifies the name of an exemption mechanism. The list of possible exemption mechanism names includes:

* KeyRecovery
* KeyEscrow
* KeyWeakening

As an example, suppose your application is exempt if either key recovery or key escrow is enforced. Then your permission policy file should contain the following:

grant {

// No algorithm restrictions if KeyRecovery is enforced.

permission javax.crypto.CryptoPermission \*,

"KeyRecovery";

// No algorithm restrictions if KeyEscrow is enforced.

permission javax.crypto.CryptoPermission \*,

"KeyEscrow";

};

NOTE: Permission entries that specify exemption mechanisms should *not* also specify maximum key sizes. The allowed key sizes are actually determined from the installed exempt jurisdiction policy files, as described in the next section.

**How Bundled Permission Policy Files Affect Cryptographic Permissions**

At runtime, when an application instantiates a Cipher (via a call to its getInstance method) and that application has an associated permission policy file, JCA checks to see whether the permission policy file has an entry that applies to the algorithm specified in the getInstance call. If it does, and the entry grants CryptoAllPermission or does not specify that an exemption mechanism must be enforced, it means there is no cryptographic restriction for this particular algorithm.

If the permission policy file has an entry that applies to the algorithm specified in the getInstance call and the entry *does* specify that an exemption mechanism must be enforced, then the exempt jurisdiction policy file(s) are examined. If the exempt permissions include an entry for the relevant algorithm and exemption mechanism, and that entry is implied by the permissions in the permission policy file bundled with the application, and if there is an implementation of the specified exemption mechanism available from one of the registered providers, then the maximum key size and algorithm parameter values for the Cipher are determined from the exempt permission entry.

If there is no exempt permission entry implied by the relevant entry in the permission policy file bundled with the application, or if there is no implementation of the specified exemption mechanism available from any of the registered providers, then the application is only allowed the standard default cryptographic permissions.