**RMJavaSDK User guide**

In order to use RMJavaSDK, you must configure your Java Runtime Environment (on the machine running RMJavaSDK) to support 256-bit keys. Due to import limits on cryptographic algorithms in some countries, the Oracle implementation of the JDK/JRE provides a default cryptographic jurisdiction policy file that limits the strength of cryptographic algorithms to a 128-bit key size, as detailed on the Oracle website on [this web link](http://docs.oracle.com/javase/7/docs/technotes/guides/security/SunProviders.html).

*Note:* As the SDK consumer, it is your responsibility to verify that you are permitted to use a 256-bit key in accordance with local regulations. Install the [Java Cryptography Extensions (JCE) Unlimited Strength JurisdictionPolicy Files](http://www.oracle.com/technetwork/java/javase/downloads/index.html) corresponding to the version of JDK/JRE you have for your machine running RMJavaSDK.

**Steps for using RMJavaSDK with Policy Controller(Deprecated)**:

1. Setup Key Management Service in Control Center. Use <CC\_Install\_Location>\NextLabs\Policy Server\tools\keymanagement\keymanagement.bat and create a Key ring named **NL\_SHARE**. After creating the keyring, create a 32 byte key in NL\_SHARE keyring. This key would be the Key Encryption Key for all encryption and decryption operations performed by any Nextlabs product connected to this Control Center.
2. Setup a Java Policy Controller and setup Key Management service plugin on the Java Policy Controller. This zip file contains the certificates and sample KeyManagement.propeties file that you need to configure on Java PC system. Remember the port number on which you have configured KeyManagement service, you will need to pass this port number as an argument to RMJavaSDK.
3. Copy the certificate i.e. key store and trust store file to the system running RMJavaSDK. These certificates will later be used to establish trust between the application running RMJavaSDK and Java Policy Controller.
4. Configure the properties file to be used by RMJavaSDK. This file should contain the following:

**KEY\_STORE\_NAME**: The absolute path of the keystore file (on the local system) required for communicating with Key Management Service. This is the same file that you have copied to Java PC while configuring Key Management plugin.  
**KEY\_STORE\_PASSWORD**: Password of the keystore file mentioned above. The sample certificate shipped with this zip has password **123next!**   
**TRUST\_STORE\_NAME**: The absolute path of the truststore file required for communicating with Key Management Service. This is the same file that you have copied to Java PC while configuring Key Management plugin.  
**TRUST\_STORE\_PASSWORD**: Password of the keystore file mentioned above. The sample certificate shipped with this zip has password **123next!**  
**PC\_HOST\_NAME**: Name or Ip Address of the Policy Controller running Key Management Service.  
**RMI\_PORT\_NUM**: Port on which the Policy Controller is running the Key Management Service.

1. Extract RMSDK-<version>.zip present inside the zip file.
2. Add KeyManagementService.jar, slf4j-api-1.7.7.jar and RMJavaSdk.jar to classpath of your project. If you are using command line, you need to compile the java sample files using javac command and mentioning the other jars in the classpath. Once you compile the java classes, you will get .class files which can be run from command line using java command.

Example:

javac -cp .;RMJavaSdk.jar RMJavaSDKAllFunctionalityTesterUsingPC.java

java -cp .;RMJavaSDK.jar RMJavaSDKAllFunctionalityTesterUsingPC

The above command assumes that all the jars are present in your current directory.

1. Refer to sample code classes in the zipped file to see how to use the SDK. The SDK has thorough javadocs also. It is highly recommended to read the javadocs if there is any uncertainty about usage of the SDK.