Izzentek Licensor Manual v1.0

This manual describes in detail the process for securing your software with Izzentek Licensor, with instructions on how to run and use each of the components of Izzentek Licensor. For a high-level overview of the licensing process, please refer to the white paper entitled *The Izzentek Licensor Process*.

Depending on what operating system you have, you will have either downloaded **licensor-windows-v1.0.zip** for Windows, or **licensor-unix-v1.0.tar.gz** for UNIX. Decompress and extract the contents of the archive into a folder of your choice. The archive contains nested archives for each of the Izzentek Licensor modules. For each module, there is a version for Java 5 and a version for Java 6. Use the archive for your target JRE version.

License Key Generator

The License Key Generator generates the cryptographic public-private key pair. The following instructions detail how to run the License Key Generator on your operating system.

On *Unix/Linux*:

1. Ensure you have acquired the proper package for your Java version (either Java 5 or Java 6). The rest of the instructions refer to the Java 5 archive name, but work the exact same way with the Java 6 archive.
2. Copy the **licensor-key-generator-java5-v1.0.tar.gz** archive into the folder in which you would like to install the software.
3. Decompress the archive by running: **gunzip licensor-key-generator-java5-v1.0.tar.gz**
4. Extract the files from the archive: **tar -xvf licensor-key-generator-java5-v1.0.tar**
5. Delete the archive once the files are extracted: **rm licensor-key-generator-java5-v1.0.tar**
6. The Java runtime engine must be on your operating system path in order for the command to run properly. If it is not set, set it now (e.g. your OS path should include the *bin* folder of your JRE installation).
7. Execute the following command to run the License Key Generator (argument details explained below): **./generate-keys.sh <*privPass*> <*privStorePass*> <*privStoreFile*> <*pubStoreFile*>**
8. As the output of the execution of the utility, you should find the public and private keystores in the location that was specified in your arguments.

On *Windows*:

1. Ensure you have acquired the proper package for your Java version (either Java 5 or Java 6). The rest of the instructions refer to the Java 5 archive name, but work the exact same way with the Java 6 archive.
2. Copy the **licensor-generator-java5-v1.0.zip** archive into the folder in which you would like to install the software.

3. Extract the files from the archive using WinZip, WinRAR, or some other Windows archiving utility.
4. Delete the archive onces the files are extracted.
5. The Java runtime engine must be on your operating system path in order for the command to run properly. If it is not set, set it now (e.g. your OS path should include the *bin* folder of your JRE installation).
6. Open a command window and execute the following command to run the License Key Generator (argument details explained below): **generate-keys -privPass <*privPass*> -privStorePass <*privStorePass*> -privStoreFile <*privStoreFile*> -pubStoreFile <*pubStoreFile*>**
7. As the output of the execution of the utility, you should find the public and private keystores in the location that was specified in your arguments.

The arguments are defined as follows:

<*privPass*> - The password that will be set for the private key. After the private key store is created, this password will be required to access the private key in the store. This password can only be at **MOST** 7 characters (or else you will get an IllegalKeySize exception).
<*privStorePass*> - The password that will be set for the private key store. After the private key store is created, this password will be required to access the store itself.
<*privStoreFile*> - The file name for the private key store that is to be generated.
<*pubStoreFile*> - The file name for the public key store that is to be generated.

Example: **generate-keys -privPass abcdefg -privStorePass gfedcba -privStoreFile PrivateStore.bks -pubStoreFile PublicStore.bks**

Once the public and private key stores have been created, they can then be used for the License Signer utility and the License SDK.

License Signer

The License Signer generates digital signatures for end-user licenses. The following instructions detail how to run the License Signer on your operating system.

On *Unix/Linux*:

1. Ensure you have acquired the proper package for your Java version (either Java 5 or Java 6). The rest of the instructions refer to the Java 5 archive name, but work the exact same way with the Java 6 archive.
2. Copy the **licensor-signer-java5-v1.0.tar.gz** archive into the folder in which you would like to install the software.
3. Decompress the archive by running: **gunzip licensor-signer-java5-v1.0.tar.gz**
4. Extract the files from the archive: **tar -xvf licensor-signer-java5-v1.0.tar**
5. Delete the archive once the files are extracted: **rm licensor-signer-java5-v1.0.tar**
6. The Java runtime engine must be on your operating system path in order for the command to run properly. If it is not set, set it now (e.g. your OS path should include the *bin* folder of your JRE installation).
7. Execute the following command to run the License Key Generator (argument details explained below): **./sign-license.sh -privPass <*privPass*> -privStorePass**

<*privStorePass*> -privStoreFile <*privStoreFile*> -signatureFile <*signatureFile*> -licenseFile <*licenseFile*>

8. As the output of the execution of the utility, you should find the license signature in the location that was specified in your arguments.

On *Windows*:

1. Ensure you have acquired the proper package for your Java version (either Java 5 or Java 6). The rest of the instructions refer to the Java 5 archive name, but work the exact same way with the Java 6 archive.
2. Copy the **licensor-signer-java5-v1.0.zip** archive into the folder in which you would like to install the software.
3. Extract the files from the archive using WinZip, WinRAR, or some other Windows archiving utility.
4. Delete the archive onces the files are extracted.
5. The Java runtime engine must be on your operating system path in order for the command to run properly. If it is not set, set it now (e.g. your OS path should include the *bin* folder of your JRE installation).
6. Open a command window and execute the following command to run the License Key Generator (argument details explained below): **sign-license -privPass <*privPass*> -privStorePass <*privStorePass*> -privStoreFile <*privStoreFile*> -signatureFile <*signatureFile*> -licenseFile <*licenseFile*>**
7. As the output of the execution of the utility, you should find the license signature in the location that was specified in your arguments.

The arguments are defined as follows:

<*privPass*> - The password required to access the private key in the private key store.
<*privStorePass*> - The password required to access the private key store itself.
<*privStoreFile*> - The file name for the private key store.
<*signatureFile*> - The file name for the file to be created that will contain the digital signature of the hash of the license file.
<*licenseFile*> - The file name of the license file for which the digital signature will be created.

Example: **sign-license -privPass abcdefg -privStorePass gfedcba -privStoreFile PrivateStore.bks -signatureFile license.hash -licenseFile license.xml**

Once the license signature has been created, it can be distributed to the end user so that they can use your software as fully-licensed.

License SDK

The License SDK is the Java library that you embed with your software so that your software can load licenses. Using the License SDK, you will be able to determine if the license is valid (there is a valid digital signature for the license), and whether or not the license was tampered with. You will also be able to determine the values of various fields specified in the license, such as the Name, Issue To, Features, and Constraints fields.

Deliverables

Your License SDK archive should be *one* of the following archives:

- **licensor-sdk-java5-trial-v1.0.zip** - Trial edition of Licensor SDK for Java 5 on Windows
- **licensor-sdk-java5-trial-v1.0.tar.gz** - Trial edition of Licensor SDK on Java 5 on UNIX
- **licensor-sdk-java5-standard-v1.0.zip** - Standard edition of Licensor SDK for Java 5 on Windows
- **licensor-sdk-java5-standard-v1.0.tar.gz** - Standard edition of Licensor SDK on Java 5 on UNIX
- **licensor-sdk-java6-trial-v1.0.zip** - Trial edition of Licensor SDK for Java 6 on Windows
- **licensor-sdk-java6-trial-v1.0.tar.gz** - Trial edition of Licensor SDK on Java 6 on UNIX
- **licensor-sdk-java6-standard-v1.0.zip** - Standard edition of Licensor SDK for Java 6 on Windows
- **licensor-sdk-java6-standard-v1.0.tar.gz** - Standard edition of Licensor SDK on Java 6 on UNIX

You will need to package and embed all the individual JARs that are in the License SDK archive along with your own software. When your software is run by the end user, these library JARs must be available on the classpath so that the License SDK can function properly. The following JARs must be embedded with your software:

- bcprov-jdk5-141.jar (or bcprov-jdk6-141.jar if you are using the Java 6 version of Licensor)
- commons-cli-1.1.jar
- licensor-common.jar
- licensor-sdk-standard.jar (or licensor-sdk-trial.jar if using *Trial Edition*)

As well, you should package the public keystore file that you generated using the License Key Generator. The public keystore needs to be supplied to the SDK when your software is run. The license and license signature can also be packaged along with the software, or they can be supplied separately.

The License File

The License File is an XML file that is described by a schema descriptor. Inside the License SDK archive, you will find the License.xsd schema descriptor which determines the structure of the license file. You will also find some sample license files that you can use as a template for your own license file. The following is a description of the various fields within the license file and what they are used for:

- **<name>**: This is the name of the license and could be the name of your software product. This field is **mandatory**.
- **<issued-to>**: This is the end-user to which this license is issued. If you have a single license for everybody, then this value could be something general like "Licensee". This field is **mandatory**.
- **<expiry>**: This is the date at which this license expires and becomes invalid. The date must be in the *MM/dd/YYYY* format (e.g. 12/31/2009 for December 31st,

2009). This field is **optional** (you should omit it if you do not want the license to ever expire).
- **<ip-constraint>**: This field allows you to specify the IP Adddress(es) that the end user's computer must be bound to in order for the license to be valid. This can be useful in preventing users from transfering licenses and their signatures to other users. This field is **optional**, and can be specified multiple times in order to specify multiple, allowable IP addresses. If no IP constraint is specified, then the license is valid for all IP addresses. The value of this field should be a string value representing the IP address (e.g. 192.16.8.0.344).
- **<feature>**: This field allows you to specify features of your software. You can deliver the same software deliverable to all your users, but restrict them from using certain features of your software based on what features they have specified in their license. For example, you can specify "Enterprise Edition" as a feature, and then using the License SDK, you can, in your software code, check for the existence of the feature in the user's license to see if your software should run in "Enterprise" mode or not. This field is **optional**, and can be specified multiple times in order to specify multiple features. The value of this field can be any string value.

When you *sign* a license using the License Signer, you are creating a separate file that contains the digital signature of the *hash* of the license file.

Using the SDK in Your Java Code

The License SDK is very simple to use. You must first instantiate a com.izzentek.licensor.sdk.Licensor object. This is the main object that you will use to load the license and retrieve license details. Next, you must load the license by calling one of the following methods on the Licensor object:

loadLicense(InputStream aLicenseInputStream, InputStream aLicenseSignatureInputStream, InputStream aPublicKeyStore)

or

loadLicense(File aLicenseFile, File aLicenseSignatureFile, File aPublicKeyStore)

If you know that you are loading the license file, the signature file, and the public keystore file from the file system, then using the second method is the easiest. The first method allows you to provide generic input streams from which you can load the license, signature, and keystore. This provides more flexibility if you want to load the data in a different way (e.g. via a network).

Note that the other methods on the Licensor object are not useful until you load the license.

When the license is loaded, the SDK validates the license at the same time. If the license signature does not match the license, an exception is thrown. Similarly, if one of the license constraints is not validated (e.g. the license is expired, or the host machine is not bound to any IP address specified in the license), an exception is thrown. It is up to your application code to handle the exception and prevent the execution of your software in a manner you deem appropriate.

If the license is loaded and validated successfully, you can use the SDK to query the features specified in the license (if any). Depending on what features are specified, you may choose have your application restrict the user from using certain aspects of the application.

The License SDK archive contains the JavaDoc API documentation (in **licensor-sdk-doc.jar**) for the License SDK, which you can use to determine all the other details of how to use the API.

Notes on Trial Version of SDK

The trial version of the SDK will identify license constraint violations by logging to the System output stream, but it will not throw any exceptions when violations are found.