Deploying IzPack

Deploying IzPack Versions to Public Repositories

General Requirements

- You must be an IzPack developer at Codehaus.
- You need a GnuPG public key.
- You need to import the Codehaus CA certificates to your local JDK

Create and Export a GnuPG Key

Your public key must be added to the **KEYS** file in the source code parent directory:

```
gpg --list-sigs "your real name" >> KEYS
gpg --armor --export "your real name" >>
KEYS
```

Export your public key to a public key server:

Example

```
gpg --keyserver search.keyserver.net
--send-key <your_key_id>
```

Without your GPG key exported to a public key server you won't be able to close a Codehaus Nexus staging repository and thus activating new releases.

In case of trouble exporting from the command line you could try submitting your key via the web UI of a keyserver of your choice, for example: http://pool.sks-keyservers.net:1137

For more information on how to create and distribute GPG keys, look at the GNU Privacy Handbook.

Configure Codehaus Maven credentials

The Codehaus parent, which relies the IzPack parent POM on, requires authentication usign the following convention in your $M2_HOME/settings.xml:

Example of settings.xml

```
<servers>
    <server>
        <id>codehaus.org</id> <!-- used for
site deployment -->
        <username>your.username</username>

<password>{encrypted_password}</password>
    </server>
    <server>
        <id>codehaus-nexus-staging</id> <!--
used for release deployment -->
        <username>your.username</username>

<password>{encrypted_password}</password>
    </server>
    <server>
        <id>codehaus-nexus-snapshots</id>
<!-- used for snapshot deployment -->
        <username>your.username</username>

<password>{encrypted_password}</password>
    </server>
</servers>
```

The server IDs are given by the Codehaus parent POM, use your Codehaus account as username and password. For encoding your personal password see the Maven documentation.

For further information see also: Codehaus Maven Repository Usage Guide.

Install the Codehaus CA Certificate into the Java JDK

1. Download the StartSSL CA and StartSSL Intermediate CA to your local filesystem.
2. Install the downloaded certificate into the JDK

If do not have permission to modify your JDK installation you can add the certificate to your own keystore in your home directory:

Example script for installing the Codehaus CA into the JDK

```
#!/bin/bash
export JAVA_HOME=/usr/java/jdk1.6.0_37
KEYSTORE=$HOME/.keystore
pushd $(dirname $0)
$JAVA_HOME/bin/keytool -import -alias
StartSSL-CA -file startssl-CA.pem -keystore
$KEYSTORE
$JAVA_HOME/bin/keytool -import -alias
StartSSL-Intermediate -file
startssl-Intermediate.pem -keystore
$KEYSTORE
popd
```

In case of using your own keystore, you must tell Maven to use it by exporting special Maven options:

Example of how to prepare the Maven call with an own JDK keystore

```
export
MAVEN_OPTS="-Djavax.net.ssl.keyStore=$HOME/.
keystore \

-Djavax.net.ssl.keyStorePassword=<your_keyst
ore_pwd> \

-Djavax.net.ssl.trustStore=$HOME/.keystore \

-Djavax.net.ssl.trustStorePassword=<your_tru
ststore_pwd"
```

For more information, see Installing a new CA into the JDK.

Deploying IzPack Snapshots

Any IzPack developer is encouraged to publish new **-SNAPSHOT** versions as they publish changes to the Codehaus Git repository. Public snapshot deployments can be done each time by a developer provided the code passes automatic tests (do not skip tests before deploying).

Test your snapshot locally by deploying it to the local repository:

```
mvn clean install
```

Deploying a snapshot to the wild is done by:

```
mvn clean verify deploy
```

Deploying IzPack Releases

Note: Before releasing, ask the fellow developers whether they agree!

For a comprehensive manual how to manage Codehaus deployments see Performing a Release.

As an example, we will consider the release of version 5.2.0.

Do a dry run

```
mvn release:prepare [--batch-mode]
-DdryRun=true \

-DdevelopmentVersion=5.2.1-SNAPSHOT \
                    -DreleaseVersion=5.2.0 \

-DpreparationGoals='clean verify install' \

-Dusername=your_codehaus_login
```

Release!

```
mvn release:clean

mvn release:prepare [--batch-mode] \

-DdevelopmentVersion=5.2.1-SNAPSHOT \
                    -DreleaseVersion=5.2.0 \

-DpreparationGoals='clean verify install' \

-Dusername=your_codehaus_login

mvn release:perform
-Dusername=your_codehaus_login
```

Don't forget to push the local POM changes and tags to Github after releasing:

```
git push --tags
```

Log-in at https://nexus.codehaus.org/, check the staging repository and if everything is fine, close and release it.

Notes:

- You won't be able to perform release:prepare without having the HTTP credentials set in settings.xml, to be able to upload files to the staging storage at Codehaus Nexus
- You won't be able to close a Nexus staging repository without having your GPG key exported to a key server.
- Things work better if your local account name matches the one at Codehaus, and you don't have to specify the *username* parameter. Otherwise, things may break, especially the Git pushes.

Post-processing and Announcing Releases

1. Use WebDAV at https://dav.codehaus.org/dist/izpack/ to push releases from the Maven repository https://nexus.codehaus.org/content/groups/public/org/codehaus/izpack
   to
   http://dist.codehaus.org/izpack/releases
2. Use the following channels for announcements:
   Website izpack.org: https://github.com/izpack/izpack.github.com, automatically deployed to http://izpack.org
   Blog news.izpack.org: Register and post an announcement. This is automatically forwarded to Twitter and Facebook.

Mailing list: user@izpack.codehaus.org, announce@izpack.codehaus.org
Google+ group IzPack (in case you're a maintainer): https://plus.google.com/105618715093492300071/posts
Facebook: http://www.facebook.com/izpack
IzPack.org website: https://github.com/izpack/izpack.github.com, automatically deployed to http://izpack.org
Twitter (ping @jponge): @izpack