

Committers Guide

Table of Contents

Build process	2
Release process	3
Prereqs	3
Release Maven Mixins	3
Release Modules	4
Recreating the archetype	4
Release an Interim Build	6
Release Snapshot	7
Docs & website	8
Prerequisites	8
Previewing the website	8
Publishing the website	9

This guide contains procedures to be performed by committers/maintainers of this codebase.

Build process

To build the platform, see the [contributors' guide](#).

Release process

This section describes the steps to release the platform to Maven central. There are four stages:

- release the maven mixins
- release the modules
- recreate the quickstart archetype (referencing the just-released modules)
- release the quickstart archetype

The release process uses Sonatype's OSS support (see [user guide](#)); our thanks to them for providing this service.

Prereqs

First, set the `$INCODEREL` environment variable to the release. Generally speaking this should correspond to the version of Apache Isis.

```
export INCODEREL=1.15.0
export INCODENEXT=1.16.0-SNAPSHOT

env | grep ^INCODE
```

Release Maven Mixins

The `release.sh` script automates the release process. It performs the following:

- performs a sanity check (`mvn clean install -o`) that everything builds ok
- bumps the `pom.xml` to a specified release version, and tag
- performs a double check (`mvn clean install -o`) that everything still builds ok
- releases the code using `mvn clean deploy`
- bumps the `pom.xml` to a specified release version

For example:

```
pushd mavenmixins
sh release.sh $INCODEREL \
              $INCODENEXT \
              dan@haywood-associates.co.uk \
              "this is not really my passphrase"
popd
```

where

- `$1` is the release version

- `$2` is the snapshot version
- `$3` is the email of the secret key (`~/ .gnupg/secring.gpg`) to use for signing
- `$4` is the corresponding passphrase for that secret key.

Other ways of specifying the key and passphrase are available, see the [`pgp-maven-plugin's documentation](#)

If the script completes successfully, then push changes:

```
git push origin master
```



The mavenmixins' `release.sh` does *not* tag the repo; this is left until the modules have also been released.

If the script fails to complete, then identify the cause, perform a `git reset --hard` to start over and fix the issue before trying again. Note that in the parent `pom.xml`, the `nexus-staging-maven-plugin` has the `autoReleaseAfterClose` setting set to `true` (to automatically stage, close and the release the repo). You may want to set this to `false` if debugging an issue.

According to Sonatype's guide, it takes about 10 minutes to sync, but up to 2 hours to update [search](#).

Release Modules

The modules are released using their own `release.sh` script. This is very similar to that of the mavenmixins; the only substantive difference is that it also tags the repo with the release version.

```
pushd modules
sh release.sh $INCODEREL \
              $INCODENEXT \
              dan@haywood-associates.co.uk \
              "this is not really my passphrase"
popd
```

If the script completes successfully, then push changes and the ta:

```
git push origin master && git push origin $INCODEREL
```

Recreating the archetype



TODO: this is still incomplete.

should post-edit the commented bits, to ensure commented out by default.

also, change the `<description>` in `incode-quickstart-archetype pom.xml`

using:

```
pushd ex/app/quickstart  
sh ../../../../scripts/recreate-archetype.sh
```

Release an Interim Build

If you have commit access to this project (or a fork of your own) then you can create interim releases using the `interim-release.sh` script.

The idea is that this will - in a new branch - update the artifacts with a timestamped version (eg `1.15.0.20170927-0738`). It then pushes the branch (and a tag) to the specified remote.

A CI server such as Jenkins can monitor the branches matching the wildcard `origin/interim/*` and create a build. These artifacts can then be published to a snapshot repository.

For example:

```
pushd modules
sh interim-release.sh 1.15.0 origin
popd
```

where

- `1.15.0` is the base release
- `origin` is the name of the remote to which you have permissions to write to.

Release Snapshot

To deploy a snapshot (to Sonatype's snapshot repo), use:

```
pushd modules  
mvn clean deploy  
popd
```

The artifacts should be available in Sonatype's [Snapshot Repo](#).

Docs & website

The website resides in the `adocs` directory:

- `documentation/` is the source for website itself (Asciidoctor)
- `template/` is the HTML template
- `search/` holds node.js Javascript files to index the built site so that it is searchable

The website is published to the [incodehq/incodehq.github.io](https://github.com/incodehq/incodehq.github.io) github repository; a `CNAME` file (in the root directory) maps this to <http://platform.catalog.org>.

To publish, this repository must also be cloned to your local computer. The scripts assume that the `incode-platform` repository (ie this repo) and the `incodehq.github.io` repository cloned at the same level, eg:

```
├── incodehq
│   ├── incode-platform
│   └── incodehq.github.io
```

Prerequisites

Make sure that you've checked out the `incodehq/incodehq.github.io` repository alongside this one (see discussion above).

You'll also need to install:

- node (v7.10.0 or later) ... used to build the search index
- python 3 ... used to preview

The actual website generation uses AsciidoctorJ, which is called by Maven plugin. There are no other software prereqs.

Normally you'll want to work in the `adocs/documentation` directory:

```
pushd adocs/documentation
```

Previewing the website

To do a quick build the website and preview locally, use:

```
sh preview-html.sh
```

This builds the HTML and the search index, but omits building the PDFs. To enable you to preview the generated site, it starts a (python) webserver to browse.

To also build the PDFs, use:

```
sh preview-html.sh
```

Publishing the website

When you are ready to publish the website, use:

```
sh publish.sh
```

This will remove all files in the `incodeh.github.io` directory and replace with the latest build.

To check everything is ok:

```
pushd ../../../../incodehq.github.io  
sh preview.sh
```

If all looks ok, then just push the changes:

```
git push
```