Release Process

Releasing KIELER

This document guides you through the steps needed for releasing a new version of KIELER.

Prerequisites

You will need a terminal application and a user account on the RTSYS servers.

Semantics Release Guide

To release a new version of the KIELER Semantics features and product, please follow these steps:

1. **Set version numbers.** Since 0.12.0 the semantics plugins and features no longer have individual version numbers. Instead all plugins share the version of the corresponding release.
   To set the version number in all pom files, manifests of all plugins and features, including their references in other files run the following script located in *build/scripts/* in the semantics repository:

   ```
   python version.py x.x.x
   ```

   

   By the way, this is a good time to make sure that every plugin has a proper license file and proper metadata. Go check the nightly RCA build to make sure that plugin names and provider names are set correctly.

2. **Check if all plug-ins to be released are contained in a feature.** This is particularly important for new plug-ins.
   Your can use the sanity check script to check the feature containment:

   ```
   python sanity.py
   ```

3. **Update the splash screen with the new version number.** Adjust the version number in the *splash.svg* in *plugins/de.cau.cs.kieler.core.product /images*. Export to *splash.bmp* (no alpha channel!) in the plugin root folder.

   Hints for splash screen adjustments:

   Dimensions: 500 x 330 px

   If your image program offers a choice, such as GIMP 2.8.10 does, it is important to save the bmp WITHOUT color space information saved or, else, the splash screen will not be displayed on Windows .... apparently a silent failure to read a "modern" BMP file (watch bug 439573). And use care, since in Gimp anyway, the choice is a negative check-off: Do not write color space information is unchecked by default, but must be checked to save the bmp WITHOUT color space information -- just like it says, but as I repeatedly misread.

   Also, The bitmap should be saved in 24-bit format (8R, 8G, 8B) for else "funny colors" (such as red instead of blue, or green instead of blue) will sometimes appear if the 32-bit format is used, with 8 bits of "transparency".
   Taken from: https://wiki.eclipse.org/Platform-releng/Updating_Branding

4. **Create a release branch in the mainline repository.** From this point on, the master branch can be used for normal development, while the release branch will only receive bugfixes and release-specific changes. Bugfixes are usually first developed on the master branch, and are then cherry-picked into the release branch.
5. **Increase the version numbers on the master branch**. After the release the master should have the version of the next release. Also adjust the splash screen. (Usually accelerates step 1 and 3 of the next release)
6. **Configure the build files in the repository.** The build instruction in the repository must be configured for the release build. To apply the configuration run the following script located in *build/scripts/*:

   ```
   python configure.py -release x.x.x
   ```

   

   This script is also capable of configuring the repository for a nightly build, to force correct configuration or revert effects of a release cofiguation.

7. **Tell Bamboo to build the release.** Configure the build plan of the last release to build the new one. Change the following settings:

   - Build Repository (corresponding release branch)
   - *semanticsReleaseVersion* Variable
   - Plan name

8. **Publish a few release candidates for testing.** Test! One aspect of testing is to assign demo videos to everyone to walk through. Make sure to create tickets for them so that everyone records his findings somewhere.

9. **Proclaim a commit freeze on the release branch.** Once all tickets are fixed, no one has any business pushing stuff into the release branch anymore!
10. **Update top-level update site.** This is done by adding the new update site URL to `compositeArtifacts.xml` and `compositeContent.xml`. These two files are in `/home/kieler/public_html/updatesite`.
11. **Create a tag of the release.**
12. **Publish release nodes and update download links in:**
    - Downloads - KIELER SCCharts Product
    - Downloads - KIELER Compiler Command-Line Interface
    - Semantics Update Site
    - Release Notes
13. **Close the Jira version, set the new default version, and add a new version.**
14. **Create a new Stream for the release in the Oomph setup.**
15. **Send the word out to the mailing list and do a release party!**

Pragmatics Release Guide

To release a new version of the KIELER Pragmatics bundles and features, please follow the following steps. They are similar to the steps for a semantics release, with some small differences:

1. **Set version numbers.**The pragmatics plugins and features share a common version number corresponding to the current release. All plugins should have the version number of one minor version higher than the previous release (see Semantic Versioning)
   To set the version number in all pom files, manifests of all plugins and features, including their references in other files run the following script located in *build/scripts/* in the pragmatics repository:

   ```
   python version.py x.x.x
   ```

   By the way, this is a good time to make sure that every plugin has a proper license file and proper metadata. Go check the nightly RCA build to make sure that plugin names and provider names are set correctly.

   There are some external bundles and features in the pragmatics repository not in the schema de.cau.cs.kieler.* that are built and hosted with a pragmatics release. Their version should stay the same as where it came from. The script does not update those fixed versions, except for the build/de.cau.cs.kieler.pragmatics.repository/category.xml file. Correct the version number manually there or correct the script to exclude that file and remove this note.
2. **Check if all plug-ins to be released are contained in a feature.** This is particularly important for new plug-ins.
3. **Create a release branch in the mainline repository.** From this point on, the master branch can be used for normal development, while the release branch will only receive bugfixes and release-specific changes. Bugfixes are usually first developed on the master branch, and are then cherry-picked into the release branch. The branch should follow this release-branch naming scheme:

   ```
   releases/pragmatics-YYYY-MM[-##]
   ```

   The optional `-##` in the end is an additional consecutive number for multiple releases in the same month.
4. **Increase the version numbers on the master branch**. After the release the master should have the version of the next release. This is important as it makes the nightly build on the master branch to be a snapshot for the next release and usually accelerates step 1 of the next release.
5. **Configure the build files in the repository.** The build instruction in the repository must be configured for the release build. To apply the configuration do the same thing as in this commit on the release branch. Make sure to update all associate sites to their release that you want to depend on.
6. **Test the Maven build locally.** Execute the maven build locally to ensure your release-branch can build successfully to avoid needing to commit multiple 'build fixed'-commits until the branch is ready.
7. **Tell Bamboo to build the release.** Configure the build plan Pragmatics Updatesite Release YYYY-MM of the last release to build the new one. Change the following settings:
   - KIELER Pragmatics Release Repository (to the new corresponding release branch)
   - *pragmaticsReleaseVersion* Variable
   - Plan name
8. **Publish a few release candidates for testing.** Test! One aspect of testing is to assign demo videos to everyone to walk through. Make sure to create tickets for them so that everyone records his findings somewhere.
9. **Proclaim a commit freeze on the release branch.** Once all tickets are fixed, no one has any business pushing stuff into the release branch anymore!
10. **Update top-level update site.** This is done by adding the new update site URL to `compositeArtifacts.xml` and `compositeContent.xml`. These two files are in `/home/kieler/public_html/updatesite`.
11. **Create a tag of the release.**
12. **Publish release nodes and update download links in:**
    - Pragmatics Update Site
    - Release Notes
13. **Close the Jira version, set the new default version, and add a new version.**
14. **Create a new Stream for the release in the Oomph setup.**

15. **Send the word out to the mailing list and do a release party!**