Papyrus Shared Components Descriptor

version 0.0.23

User Guide

Eclipse **2017-05-30**

Table of Contents

©2017, Eclipse • ALL RIGHTS RESERVED.

©2017, Eclipse • ALL RIGHTS RESERVED.

1 User

1.1 Download

The Papyrus Components guide could be downloaded as a pdf here

1.2 Papyrus Components

1.2.1 Context

The Papyrus Components project aims at providing a shared developer environment to help to integrate additional Papyrus Component.

1.3 Standard Operating Procedure

- SOP Project Component Creation
- SOP Target Platform Update
- SOP Release of a component
- SOP Add Missing License to java files
- SOP Add a Gerrit Contributer
- SOP Add Documentation to your project

1.4 Tips

©2017, Eclipse • ALL RIGHTS RESERVED.

2 SOP Component Release

2.1 sop-release

2.1.1 Who is in charge of?

Committer + a basic developer in order to transfer knowledge and improve the process.

2.1.2 When?

when you need it, on a regular basis, every two month.

2.1.3 How long?

It last at least 120mn

2.2 Prerequisite

2.2.1 Access

Have the Hudson access to the Component tab and a committer with you to make the review and merge.

2.2.2 Green

No critical bugs open for the release you are targeting.

Jobs (master,website,quality,deploy) are green

2.2.3 Version

Initial version of your pom.xml, MANIFEST.MF and category.xml should match qualifier and -SNAPSHOT

2.2.4 Change log

You must fill the changelog to describe the new version feature.

```
/org.eclipse.papyrus-sysml/src/changes/changes.xml
```

Details https://maven.apache.org/plugins/maven-changes-plugin/changes.html

2.3 Pre-Actions

You can send an email to the developers mailing list.

2.4 Steps

2.4.1 Create a bugzilla ticket.

Bug XXXXXX - [YourComponent][release][YourTargetVersion] Release

2.4.2 Upgrade your target platform

Your initial platforms are perhaps using a version setted to 0.0.0. To release your project we advice you to replace this 0.0.0 by the version of the day.

Execute the following command on the target platform you are matching:

```
mvn org.eclipse.tycho.extras:tycho-version-bump-plugin:update-target -Dtarget=/home
```

It should replace all 0.0.0 version by a strict identified version of all needed features.

2.4.3 Upgrade the version of application - version 1

Remove the .qualifier, -SNAPSHOT extension in the different pom.xml, manifest.mf artifact version.

Example of command to upgrade pom.xml, plugin.xml, manifest.mf, category.xml, feature.xml etc..., check if it runs on all plugins depending on your profile configuration

```
mvn org.eclipse.tycho:tycho-versions-plugin:set-version -DnewVersion=1.2.2.qualifie
```

2.4.4 Upgrade the version of application - version 2

Go to the root pom.xml

Update the version (removing the -SNAPSHOT if necessary) of the target.version

Go to the targetPlatform folder

Launch the following maven command using the new version as X.Y.Z (and removing the -SNAPSHOT if necessary):

Use tycho-versions plugin to switch from qualifier to release, and then back from release to qualifier

```
mvn org.eclipse.tycho:tycho-versions-plugin:set-version -DnewVersion=X.Y.Z-SNAPSHOT
```

Then verify the build locally using the following command

```
 mvn clean install -Dtycho.localArtifacts=ignore
```

Update the plugin versions by finding and replacing manually the former X.Y.Z-SNAPSHOT and X.Y.Z.qualifier to the new X.Y.Z version. A command can be used, such as at the root:

```
 find . -type f -name "file" -exec sed -i 's/X.Y.Z-SNAPSHOT/X.Y.Z/g' {} +
```

check the different pom.xml, feature.xml, category.xml and MANIFEST.MF. Those can be checked either manually or using the following command:

```
 find . -name "file to test" | xargs grep -n -e "qualifier" -e "SNAPSHOT"
```

Push on gerrit the different modifications

Check the status of the gerrit job

Add a reviewer

Review and merge the change

©2017, Eclipse • ALL RIGHTS RESERVED.

2.4.5 Rexecute the job Master and eventually job Website

Who: any

Goal is to use this job version as data for the promotion.

2.4.6 Make the release

Who: committer action

Go the Hudson Component tab

Execute the job papyrus-component-deploy-eclipse

Fill the args

It should tag the release automatically, you have to check it by looking at the git repository

It should also automatically execute the job papyrus-component-deploy-nexus: to deploy the artifact into the Eclipse Papyrus official Nexus

It should also automatically execute the job and papyrus-component-deploy-website

2.4.7 Deploy the web site

Who: non-committer action + committer review

The new web site is available under the target/site-staging directory

Download it as a zip

Unzip it in the Papyrus web git repository, under the papyrus/components/YourComponent directory

Update the root index.html with the new version of your component

```
https://git.eclipse.org/c/www.eclipse.org/papyrus-sysml.git/
```

Ask for validation for the review.

Once it is merged, you should see it 3mn here http://www.eclipse.org/papyrus/components/MyComponent/YourTargetVersion/

2.4.8 Upgrade to the next snapshot version of the application

use tycho-versions plugin, use the good profile to execute it

```
mvn org.eclipse.tycho:tycho-versions-plugin:set-version -DnewVersion=X.Y.Z-SNAPSHOT
```

check the different pom.xml and MANIFEST.MF, feature.xml and category.xml files

You have eventually to manually update the category.xml at /org.eclipse.papyrus-sysml/releng/org.eclipse.papyrus.sysml14.p2/category.xml

Push it as new patch and make the review.

2.4.9 Restore eventually the 0.0.0 in the target platform

It could be good to restore the 0.0.0 version in your future target platform

2.5 Post-Actions

Close the initial ticket.

You can send an email to the developers mailing list

©2017, Eclipse • ALL RIGHTS RESERVED.

3 SOP Add Missing License

3.1 Context

Sometimes your gerrit failed due to missing license header files. This has been detected thanks to the com.mycila.license-maven-plugin plugin

3.2 How to add license in batch mode?

```
mvn license:format -Dlicense.header=/home/flefevre/gitNeon/org.eclipse.papyrus.tool
```

3.3 Run mvn from any pom directory and enable license check

Add the following property to reference the root directory of the sysml project

```
-Dcomponent.root=
```

4 SOP Target Platform Creation

4.1 Context

All components are based upon a set of targets platform localized at the targetplatform directory. You can update them with the Obeo plugin or directly from a maven command.

It will switch resolve all feature with the latest ones found on the different update site you are referring.

4.2 How to?

4.2.1 Pre-requisite

You have to have a target.file at the root of your target platform plugin. You need to specify with target platform to activate through profile management.

```
mvn org.eclipse.tycho.extras:tycho-version-bump-plugin:1.0.0:update-target -Dtarget
```

the tycho-version-bump-plugin is bound to the validate phase

-DtargetUpdate=true ensure you enforce the taregtUpdate

eventually add -Declipse.targetrelease=neon-papyrusnightly -Declipse.release=neon-papyrusnightly: specify with target to update

and -f targetplatform/pom.xml: specify the pom to look at

©2017, Eclipse • ALL RIGHTS RESERVED.

5 SOP Add Documentation to your project

5.1 Several ways to add documentation to your project

A good project is a project with uptodate documentation. To help you in this task, Papyrus components has pre-configured several ways to add documentation.

Your documentation could target the developers of your product and also the user of your product.

To ease the way the documentation is managed, Papyrus components has taken the decision to hold it only in the source code.

Please consider the following ways to add a documentation to your project

- Eclipse embedded documentation, the help content
- Papyrus components web site

5.2 Papyrus components website

All documentation at the maven formats (fml, markdown,pdf,xdoc) localized under src site will be processed to generate a dedicated website

Please consider the sub tree folders

- user: people that will use your product
- developer: people that will develop your product
- relenger: committer that will be in charge of making the release of your product

By convention, Papyrus components will look at the following pdf files

- pdf user-guide.pdf.xml
- pdf developer-guide.pdf.xml
- pdf relenger-guide.pdf.xml

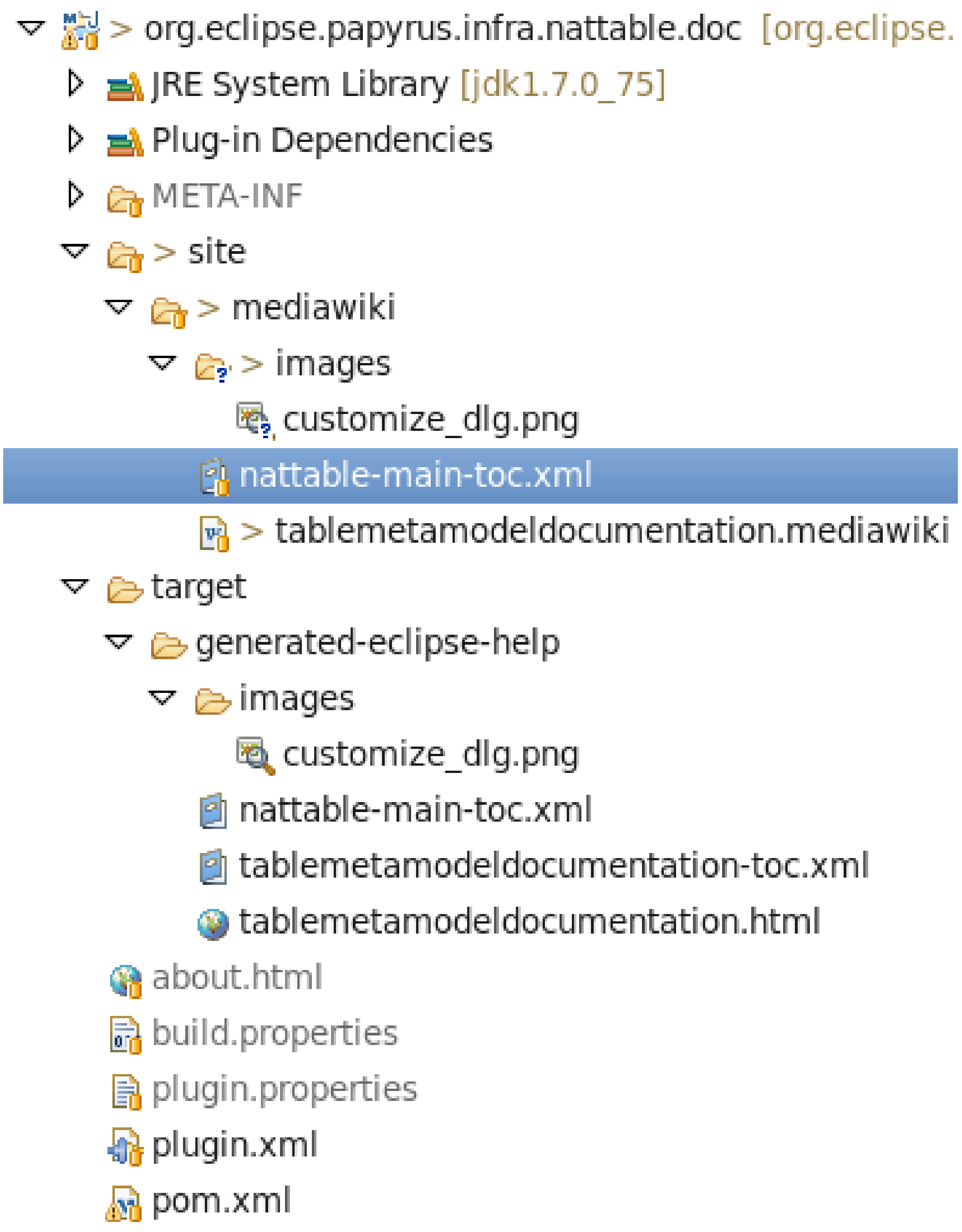
5.3 Eclipse embedded documentation

5.3.1 Context

Documentation should be accessible online and in the documentation embedded in Eclipse platform. Both places should be coherent, so a common format is used and mechanism of synchronization should be used with a main repository. Documentation should follow these recommendations Eclipse Doc Style Documentation is localized inside several plugins under org.eclipse.papyrus/plugins/doc. The official version is the one attached to the git repository and release with the Papyrus feature/ plugins. Please be aware that there is a discussion on releasing the developer documentation directly from the git repository through the usage of maven site plugin. If you have any idea or requirements, do not hesitate to contact us through the developer mailing list. The proposition is to maintain the documentation only in the source code repository and to have a mechanism to publish it directly in the Eclipse web site and in the Eclipse Papyrus product. Pending discussion...

5.3.2 Add your own documentation in the plugin

Here a snapshot of a typical documentation plugin. Notice that the target directory is a generated directory.



```
▽ > org.eclipse.papyrus.infra.nattable.doc [org.eclipse.
  ▷ JRE System Library [jdk1.7.0_75]
  ▷ Plug-in Dependencies
  ▷ META-INF
  ▽ > site
    ▽ > mediawiki
      ▽ > images
          customize_dlg.png
        nattable-main-toc.xml
        > tablemetamodeldocumentation.mediawiki
  ▽ target
    ▽ generated-eclipse-help
      ▽ images
          customize_dlg.png
        nattable-main-toc.xml
        tablemetamodeldocumentation-toc.xml
        tablemetamodeldocumentation.html
    about.html
    build.properties
    plugin.properties
    plugin.xml
    pom.xml
```

calling the MyLyn mediawiki maven plugin in the pom.xml You plugin will contains a site directory that will hold your mediawiki documentation with it embedded resources such as images. The mediawiki files will be processed by a maven wiki plugin. Here some details of the configuration.
It will generate a toc file and the corresponding html file in the output folder. If you want more information, please have a look to MediaWiki

```
<plugin>
<groupId>org.eclipse.mylyn.docs</groupId>
<artifactId>org.eclipse.mylyn.wikitext.core.maven</artifactId>
<configuration>
<sourceFolder>site/mediawiki</sourceFolder>
<outputFolder>${project.build.directory}/generated-eclipse-help</outputFolder>
<copyrightNotice>${help.copyrightNotice}</copyrightNotice>
<title>${help.documentTitle}</title>
<multipleOutputFiles>false</multipleOutputFiles>
<navigationImages>true</navigationImages>
<formatOutput>true</formatOutput>
<htmlFilenameFormat>$1.html</htmlFilenameFormat>
<xmlFilenameFormat>$1-toc.xml</xmlFilenameFormat>
<helpPrefix>target/generated-eclipse-help</helpPrefix>
<stylesheetUrls>
<param>styles/main.css</param>
</stylesheetUrls>
</configuration>
<executions>
<execution>
<goals>
<goal>eclipse-help</goal>
</goals>
</execution>
</executions>
<dependencies>
<dependency>
<groupId>org.eclipse.mylyn.docs</groupId>
<artifactId>org.eclipse.mylyn.wikitext.mediawiki.core</artifactId>
<version>${mylyn.wikitext.version}</version>
</dependency>
</dependencies>
</plugin>
```

Create a new TOC file that will reference the generated TOC file Content of the myplugin-main-toc.xml file :

```
<?xml version='1.0' encoding='utf-8' ?>
<toc label="Search" link_to="../org.eclipse.papyrus.infra.doc/toc.xml#PapyrusDocUse
   <topic href="target/eclipse-generated-help/myplugin.html" label="Search in Model
      <link toc="target/eclipse-generated-help/myplugin-toc.xml"/>
      <anchor id="searchInModel"/>
   </topic>
</toc>
```

link_to add the contribution to the PapyrusDocUser anchor (defined in the main papyrus documentation plugin) warning: paths are related to the root of the plugin the generated TOC is referenced by the hand-written one. An anchor is also added here, so the search documentation can be extended by another contribution. Add all images embedded in the documentation in the same folder as the mediawiki and all generated files. Do not forget to add all files (the folder resource in the example) to the build.properties file. Reference the TOCs in the extensions of the plugin

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension point="org.eclipse.help.toc">
      <toc file="target/mediawiki/main-toc.xml" primary="false"/>
   </extension>
</plugin>
```

6 FAQ

6.1 User Frequently Asked Questions

General

1. How to clear the different caches?

6.2 General

How to clear the different caches?

You have several options:

Clean the local project artifacts

```
mvn clean install
```

Force update of the downloadded plugins

```
mvn install -U
```

Remove manually downlaoded artifacts, for example

```
rm -rf .m2/repository/org/eclipse/papyrus
```

Ignore local tycho artifacts

```
mvn clean install -Dtycho.localArtifacts=ignore
```

Force the cache of tycho

```
rm -rf .m2/repository/.meta/p2-local-metadata.properties
```