## Table of Contents

- Publishing conventions for custom publisher ................................................................. 1
  - Location ..................................................................................................................... 1
  - Version and build architecture .................................................................................. 1
  - RPM Packaging Attributes ..................................................................................... 1
  - RPM Documentation Attributes ............................................................................. 2
Publishing conventions for custom publisher

(recipe provided by Alberto Di Meglio)

In order to publish packages using the ETICS Publisher the following conventions have to be followed by the Packager (the ETICS Packager follows these conventions, which means you get them for free when using it):

**Location**

The build scripts must put the tarballs in a directory called tgz in the root of the module and the (S)RPMS in a directory called RPMS in the root of the module. The internal structure of the tgz and RPMS directories is not important since tarballs and rpms are searched within the tree starting from those directories.

It is also possible to redirect the location where ETICS looks for packages by defining the following properties:

- `package.tgzLocation` (for tarballs)
- `package.RPMSLocation` (for binary RPMS)
- `package.SRPMSLocation` (for source RPMS)

**Version and build architecture**

When the packages are published by ETICS, they are put in the path:

`${workspaceDir}/dist/${projectName}/${moduleName}/${version}/${platform}`

The version string is taken from the metadata of the configuration object, which must therefore match the package version. If the packages are created by ETICS this is guaranteed by definition, otherwise you have to make sure that your packaging script use the correct version information.

The `platform` string is taken from the detected platform information for platform-dependent packages and from the `$buildarch` property in other cases. If your packages are not platform-dependent, make sure you attach to the configuration the property:

`buildarch = noarch`

Following these simple rules, the ETICS packager will harvest your packages and publish them automatically, as part of the `publish` target. Remember that the `prepublish` / `postpublish` targets are also available should you require any adjustments before and/or after publishing.

-- MebSter - 11 Dec 2006

**RPM Packaging Attributes**

- `package.userspec=(path to the RPM spec file relative to the module directory)`
- `package.customtag`
- `package.group = (refers to RPM's Group: tag)`
- `package.packager = (refers to RPM's Packager: tag)`
- `package.prefix = (/opt/glite by default in the org.glite project)`
- `package.buildarch = ('noarch' for example for script components)`
- `package.requires = (refers to RPM's Requires: tag)`
- `package.provides = (refers to RPM's Provides: tag)`
• package.obsoletes = (refers to RPM's Obsoletes: tag)
• package.Conflicts = (refers to RPM's Conflicts: tag)
• package.autoreqprov = (refers to RPM's AutoReqProv: tag, should be 'no', when any of the above ones used)
• package.configure
• package.clean
• package.compile
• package.doc
• package.install
• package.preserve.libtool

Unclear of the exact name: preun, postun, post

**RPM Documentation Attributes**

There are two ways of setting the RPM 'Summary' and 'Description' fields:

- description = becomes the RPM 'Summary' value
- moduleDescription = the long description of the component goes to 'Description'

or directly via

- package.summary = RPM's Summary
- package.descriptio = RPM's Description

Unclear of the exact name: changelog

-- AkosFrohner - 15 Mar 2007