Table of Contents

EMI Packaging Policy

Latest approved version of this policy
11.03.2011: EMI_SA2_Packaging_v_1_0.pdf

|

Introduction

This document describes the EMI policy to be followed when creating packages for an EMI software component.

Definitions

- **Component**: It refers to any of the EMI software components. The list of EMI software components can be found in Section 4 `EMI Components and Product Teams`, page 11 of DNA1.3.1 - Technical Development Plan.
- **Binary packages**: executable packages.
- **Source packages**: package files that contains everything needed to recreate not only the programs and associated files that are contained in the binary package file, but the binary and source package files themselves.

Package Formats

Product Teams must support all package formats of the EMI supported platforms.

Since EMI-1 is supported on SL5 x86_64, the mandatory package formats are: **tar.gz, src.tar.gz, rpm, src.rpm**

Buildable source packages must be produced.

Since EMI-1 is supported on SL5 x86_64, all source packages (src.rpm) will be validated by a Mock build in a pristine environment.

Metapackages

Product Teams must provide meta-packages to define sets of packages to be installed together.

The metapackage names must start with the string `emi-` and then have the rest of the name in LOWECASE (i.e. `emi-voms`). The LOWERCASE is mandatory in the Debian packaging policy.

The following rules must be followed for meta-packages:

- The meta-package must be empty. No files included.
- The meta-package must include only first level dependencies (that is the minimal set of packages allowing the complete installation of the product).
- The meta-package must not set any version constraints for the dependencies
- The meta-package version must not change when any new versions of its dependencies are released. A new meta-package version is released only when dependencies are added or removed.

NOTE: Metapackages do not allow the installation of 32bit packages on 64bit boxes and therefore are not suitable for packages (i.e. clients) that have this requirement.

ETICS Package configuration

- Each project configuration can use the property `package.repo` to specify a YUM repo file from where pick the EMI packages produced during the nightly builds.
- The `package.autoreqprov` must be set only in project configuration and its value must be `yes`.

- The `package.prefix` property must **NOT** be set by components. This property is set at project level as `/`.
- When using the ETICS Pakager:
  - In the install target of the build commands, install the files in the ${prefix} directory as if ${prefix} would be /
  - The ETICS Packager removes the Prefix set to `/` making the package non relocatable.
- When not using the ETICS Packager:
  - Make sure not to have the Prefix specified in your specfile
  - Make sure to produce a tar.gz matching the content of the RPM (therefore including the full path /usr, /usr/bin, etc.)

Examples of SPECFILE (show) Examples of SPECFILE (hide)

- SPECFILE: dcap
- SPECFILE: VOMS
- SPECFILE: root

Package Contents and Metadata

The EMI project has adopted the well known packaging guidelines and policies defined by Fedora, EPEL and Debian.

Please refer to:

- Fedora Guidelines and Policies
- EPEL Guidelines and Policies
- Debian Policy Manual

The EMI project adopts the acceptance criteria of such documents to validate packages on these specific platforms.

ETICS provides the RPMLint plugin to verify that the RPM generated complies with the Fedora Guidelines.

Build in pristine environments

- From EMI 1 RC1 it is possible to build in pristine environments leveraging the Mock integration of the ETICS client 1.5.0 and the SUDO capability of the ETICS worker nodes.
- This is possible by adding the option `--repackage=[MOCK_CONFIGURATION]` in the etics-build command.
- The option is enabled only if the user running the build has root or sudo privileges.
- This option collects all the src.rpm generated during the normal ETICS build and executes Mock in a pristine environment rebuilding all packages one by one.
- This repackaging happens at the end of the ETICS normal build.
- In addition to the OS and EPEL YUM repositories, the Mock build makes use of a locally generated and EMI RCX YUM repositories.
- In the Package List section of the report, it is possible to see what packages have been built successfully via ETICS and/or Mock.

- From where the dependencies are installed:
  - Dependencies which are part of the same build, get built before and are then installed from the locally generated YUM repository.
  - Dependencies part of EMI but which are not part of the same build, get installed from the current nightly YUM repository (defined in `package.repo`).
  - External dependencies are taken from EPEL or OS YUM repositories

Contacts

- Lorenzo.Dini@cernSPAMNOT.ch

Logbook

v1.0 (approved on 11.03.2011)

- *11 February 2011*: First draft prepared by Lorenzo.
- *11 March 2011*: Maria applies feedback given by PEB.

This topic: EMI > EmiSa2PackagingPolicy
Topic revision: r16 - 11-Mar-2011 - 11:39:18 - MariaALANDESPRADILLO