
# -WP4 Workshop-
# Developers' Guide

**Maite Barroso**

**10/12/2002**

**Maite.Barroso.Lopez@cern.ch**

# What is the developers' guide?

- This guide provides guidelines for developing code for the European DataGrid project. It is intended to be the definitive source of information for everything which the developer needs to know.
- It has been constructed with information taken from a number of other document which it supersedes. These include:
  - initial Developer's Guidelines
  - Building Conventions
  - DataGrid Naming Conventions
- The Quality Group is responsible for updating/maintaining it.

# Enforcement of the proposed guidelines

- It is realised that some of the obligatory rules cannot be enforced immediately for existing code.
- However, the QAG expects to see a shift towards following the obligatory rules and expects that all new code will conform as it is written.
- Random cross-checks of code and documentation done by QAG members

# Guide Skeleton

- Software Packages
- Automatic Build System
- Environment
- Interfaces and APIs
- Documentation
- Code management
- Test and Validation process
- Style and Naming Conventions
- Integration Procedure

# Software Packages

- Developers must provide a complete, accurate list of the dependencies for each package.
- When external packages are needed, developers must check if they are already inthe external packages section of the DataGrid repository. If not or if a different version is needed, the developer must contact the Integration Team to discuss possible conflicts with other work package requirements. Once there is agreement the package will be added to the repository.
- The packaged code must be entirely relocatable via the environmental variable EDG_LOCATION

# Automatic Build System

- Packages are created from tagged versions of CVS modules. For each package version x.y.z there must be a corresponding tag vx_y_z of the CVS module.
- Automatic Build on demand (BOD) is trigered when:
  - A regular release tag (like v1_0_2) is set; the built RPM will immediatly be published.
  - A special "build" tag of the form build[0-9]*, that is the string build followed by any arbitrary number. The produced RPMs will not be published in that case.

# Environment

- EDG software services shall source/parse a top level EDG configuration file, and shall not rely on this being previously done
- This file will have a key-value based syntax, containing all the needed basic prefixes:
  - EDG_LOCATION /opt/edg
  - EDG_LOCATION_VAR $EDG LOCATION/var
  - EDG_TMP /tmp
- Daemons:
  - should not run as root
  - All initialization of a daemon should be done with the init.d script which controls the daemon. The init.d script should minimally support the start, stop, status, and restart methods.

# Interfaces and APIs

- API guidelines:
  - The API should be managed as a separate package whenever possible. This is to ensure that the code is not tied in any way to the implementation. The interface package should be versioned independently of any implementation. This makes it very clear when an API changes.
  - For web and grid services, language specific bindings must be derived from a WSDL document whenever possible. This allows the functionality of a service to be defined in a language independent manner.
- Interface versioning: guidelines to maintain versions of interface packages.

# Documentation

- README and INSTALL files
- API documentation should be provided with Doxygen for C/C++ code, JavaDoc for Java code and POD for Perl.
- Recommended source formats for end-user documentation include sgml/XML, html and LATEX.
- There must be a man page for each executable and each configuration file. Those man pages have to be installed under /opt/edg/share/man by default.

# Code Management

- Guidelines on version numbering: Version numbers should be in the format version.revision.patch flavour, e.g. 1.0.2.
- Organization and access to the cvs repository
- Organization and access to the package repository

# Test and Validation process

- This section describes the release process as a cycle of well defined every day activities (development-integrating-testing), application testing and the system certification leading to the next software release:
  - WP development and unit tests
  - Integration
  - Application validation
  - Testbed structure

# Style and naming Conventions

- It contains basic Java and C++ coding and naming conventions

# Integration Procedure

- Detailed description of the integration procedure:
  - What needs to be provided before the integration starts (autobuilt RPMs, automatic configuration, documentation, unit tests)
  - The package will be built from CVS with the autobuild system. If it doesn't build, the integration for this package ends. The work package must contact the technical coordinator to schedule a new slot for integration.
  - If the unit tests fail, the work package must immediately correct the problem and provide a new tagged package. If the problem cannot be corrected immediately, the integration for this package ends and the work package must schedule a new slot with the technical coordinator.

# Present status

- The official first version is not out yet. It is presently being reviewed by QAG and thecnical coordinator.
- It will be made public before Christmas.
- The latest version is:
  - https://edms.cern.ch/document/358824/0.3
- Please, read it and apply it
- send me your comments if you strongly disagree with any guideline