Journal of Physics: Conference Series **219** (2010) 042051 doi:10.1088/1742-6596/219/4/042051

CMS Partial Releases: Model, Tools, and Applications Online and Framework-Light Releases

Christopher D. Jones, David Lange, Emilio Meschi, Shahzad Muzaffar, Andreas Pfeiffer, Natalia Ratnikova[1], Elizabeth Sexton-Kennedy

Institut für Experimentelle Kernphysik, KIT, Wolfgang-Gaede-Str. 1
76131 Karlsruhe, Germany

Email: Natalia.Ratnikova@cern.ch

Abstract.With the integration of all CMS software packages into one release, the CMS software release management team faced the problem that for some applications a big distribution size and a large number of unused packages have become a real issue. TWe describe a solution to this problem. Based on functionality requirements and dependency analysis, we define a self-contained subset of the full CMS software release and create a Partial Release for such applications. We describe a high level architecture for this model, and tools that are used to automate the release preparation. Finally we discuss the two most important use cases for which this approach is currently implemented.

1. Introduction

The Offline software system [2] of the CMS experiment [1] comprises more than thousand packages organized in subsystems for analysis, event display, reconstruction, simulation, detector description, data formats, common framework, and other utilities and tools which are all maintained within one common project CMSSW. Each CMSSW release depends on about a hundred external products, which we also build from source and package for distribution. In order to manage this complex system efficiently the CMS Software Development Tools group has been working on the automation of the software release integration and distribution procedures using a range of tools [3] developed within or adopted by CMS. Thus, CMS has adopted the RPM Package Manager [4] technology for the software packaging, and has chosen the Advanced Packaging Tool, APT [5] for the package management and distribution.

Briefly, the release procedure consists of the following steps. First, the content of the release is defined. All CMSSW packages and their respective versions are defined in the CMS Tag Collector [6]. External products, their versions, configurations, and build instructions are defined in the CMSDIST [7] repository. Next, every external package is built and packaged in rpm format. Then the whole CMSSW release is configured, built and packaged into one rpm file. Finally, all resulting rpm

[1] To whom any correspondence should be addressed.

files are uploaded into a central repository for further distribution and deployment on the CMS computing resources.

Alongside this generic approach there are specific use cases where only limited subsets of packages and externals are required.

The two most important examples are:

- High Level Trigger (HLT) algorithms running on the online Event Filter farm [8].
- CMS lightweight interactive physics analysis tool, Framework-light[9].

Details of these use cases will be considered later in Section 5. In the next two sections we present the model of Partial Releases and the tools that allow to build customized releases for the specific applications. In Section 4 we describe the implementation of the Partial Releases and how we address the two main technical challenges: consistency with the full release, ensured by construction, and the automation of all steps of the procedure for optimal support at reduced maintenance cost.

2. Concepts

The model of partial releases is based on the following concepts:

- *Base Release* – is a given version of a standard CMS Software release, including the full set of CMSSW packages, and a configuration of external software products, built, packaged and distributed using standard CMS release management tools.
- *Application Set* - is a subset of packages of the base release that directly provides the desired functionality.
- *Build Set* - is a complete minimal subset of packages and external products of the Base Release, required for building all packages of the Application Set.
- *Partial Release* - is an application oriented independent software release of packages defined in the Build Set.

The Partial Release is built and packaged using standard CMS release management tools.

3. Tools

The procedure of creating a Partial Release relies on the following tools developed within CMS:

- *Ignominy* [10] is a powerful general purpose tool used to analyse the source code, binaries and other contents of the software release, in order to detect different types of dependencies between software packages.
- *buildset* provides a simple interface that allows one to query the output produced by Ignominy, and recursively calculates the list of dependencies for a given package or a list of packages.
- *CMSDIST* is a repository of specifications files that contain instructions for the product configuration, build, packaging, installation, and setup. Here, each external package, the full CMSSW release, and each partial release is considered as a separate product.
- *PKGTOOLS* is a framework for building, packaging and uploading software distributions according to instructions in the CMSDIST.
- *Tag Collector* is a central interface for adding new versions of packages to the release and managing the contents of the CMSSW release on a package level.
- *SCRAM* provides the CMS software build and configuration management system and user interface.

4. Implementation

As illustrated by Figure 1 below, the Base Release represents a large number of interdependent packages. We define a subset of these packages that provides a desired functionality for a particular application and call it Application Set. Some packages included in an Application Set may depend on another package of the Base Release which is not a part of the Application set. All packages included in or required by the Application Set and all corresponding external products constitute a Build Set for the Partial Release.

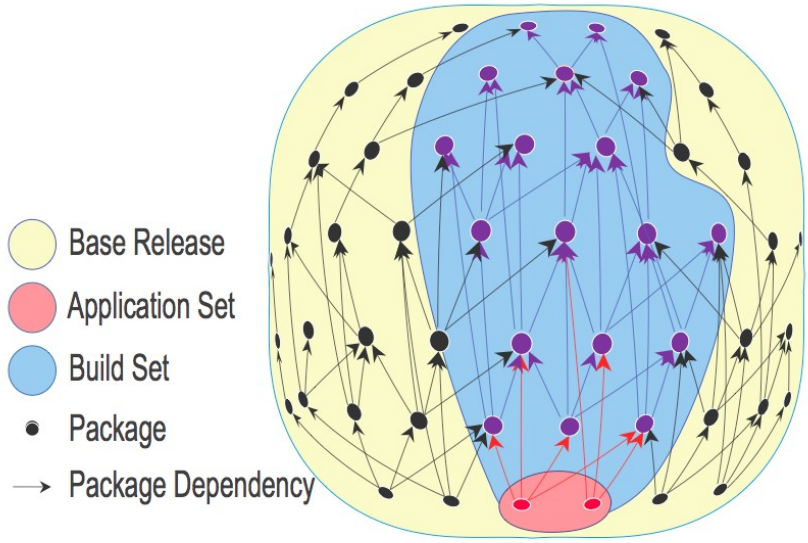

Figure 1 : Illustration of the partial release model.

The procedure of building a Partial Release starts from the request of the application manager. Based on the functionality requirements he or she defines the base CMSSW release and the corresponding Application Set. All the subsequent steps are performed by the release manager.

The crucial point of finding the correct Build Set is the discovery of the dependencies between the packages. These dependencies may occur at different levels. For example, source code dependencies happen at compilation time, primarily via included header files. Binary dependencies occur at link time via symbols in the libraries. Run time dependencies may happen via dynamically loaded plug-in modules. All dependencies need to be satisfied for the package to build and run successfully.

In order to extract and collect all necessary dependency information for the Base Release, we run Ignominy. The Ignominy analyzer examines the source code, the binaries (executables and libraries), and configuration files available in the installation of the Base Release and corresponding external packages and produces a simple dependency database in a form of an ASCII text file. This database and the accompanying log file include the details about each individual source of dependency identified.

Next, we run the *buildset* script which takes as input the Application Set and the results of the Ignominy analysis for the Base Release. The *buildset* recursively analyzes the package dependencies and discards any superfluous connections such as extra dependencies introduced in the local unit tests. The resulting output contains the final self-contained list of packages and external products for the Partial Release.

Note, that in this procedure we ensure by construction that the Partial Release only includes those packages that are either required by functionality or come as a necessary dependency via the Build Set.

Once the Build Set is defined, we start building the release. As mentioned before, package build and configuration rules are defined in the specification files stored in the CMSDIST repository. The

specification file for the Partial Release is constructed in such a way that it first downloads the source code distribution of the corresponding Base Release. Then, all packages included into the Build Set are selected and copied into the build area of the partial release. Thus, the Partial Release will automatically reuse the source code of the Base Release, but only packages included into the Build Set are built.

The list of external products for the Partial Release is maintained separately. This list only includes the names of products, without specifying their versions. The consistency on the version level is enforced by the PKGTOOLS framework [7].

Build and configuration rules for CMSSW packages are defined in the shared area in CMSDIST. Partial Release uses the same set of rules as the base CMSSW release does. In this implementation the overall consistency of the code, external products, and common build rules used in the Partial and Base releases, is ensured by construction.

5. Examples of Applications

The concept of CMS Partial Releases was driven by two major use cases: the software distribution of the HLT algorithms running on the online farm, and the distribution of the lightweight CMS software framework for the interactive physics analysis in the ROOT environment.

5.1. Online Release

The goal of the Online Release [10] is to provide the software environment for the HLT online operation. A general requirement to the online environment is that it must be consistent with the Offline software environment. It is also important to minimize the amount of code to be deployed on the online farm.

Additional requirements [11] were identified due to a special nature of the online computing environment:

- Minimal amount of code to improve robustness and stability of the online operations.
- Possibility to apply specific compiler options for optimizations, hence different architecture name and completely separate set of binaries.
- Possibility to reuse packages already available on the local system, such as online version of data acquisition software Xdaq [12], and corresponding external products.
- Software distribution must be suitable for use with the Quattor tool [13], which is used to manage software deployment on the online farm, and has a number of constraints with respect to the regular CMSSW installation method based on APT.
- We should also offer the possibility of producing patch releases forking from the mainstream CMSSW release sequence to retrofit necessary fixes while guaranteeing stable operations of the online HLT.

The online release Application Set includes HLT reconstruction packages, filter modules, data acquisition, and Data Quality Monitoring tools. A special effort has been applied to make sure that none of the packages included into the Online Application Set would bring in a dependency on detector simulation and physics analysis packages. The configuration has been adjusted in such a way that required external products, which are available in the system installation including compiler, XDAQ, and some other packages, would be used in place of versions normally distributed with CMS software.

The rpm packages for all remaining external products and the Online release itself are then built and named according to the conventions used in the Online system according to requirements imposed by the use of Quattor.

To enable a possibility of quick updates of the HLT modules in case of urgent corrections, an additional procedure of patch releases has been developed. The patch release distribution only includes new versions of packages specified by the Online application manager in a dedicated release queue of the CMS Tag Collector. The rest of the packages are distributed with the original "parent" release.

5.2. Framework-light release

The goal of the Framework-light application is to provide the possibility of analyzing CMS physics data in an interactive ROOT [14] environment on the user's laptop. CMS Analysis Object Data (AOD) are stored directly in ROOT format, hence access to these data from the interactive analysis environment is natural provided that all necessary libraries for the data classes are available. Consequently, Framework-light Application Set contains the following:

- ROOT and external packages required by ROOT
- All necessary CMS Data Format packages
- Tools and algorithms for the interactive physics analysis

A special effort was made to eliminate any accidental dependencies that would bring additional unneeded packages into the Framework-light distribution. A small distribution size allows easy download and installation of the Framework-light distribution onto the user's laptops. Minimal number of external dependencies and small amount of code simplify porting to other platforms. Currently Framework-light runs on all basic Linux distributions and on the MacOS platform.

Modest size and easy access to CMS data structures, combined with powerful ROOT visualization tools, make the Framework-light tool attractive and popular among the physicists.

6. Summary and Conclusions

The model of Partial Releases enables us to build and distribute customized software releases for a number of well-defined applications where the large size and complexity of the standard full CMSSW distribution become an issue. For illustration, in a full CMSSW Base Release including 1114 packages and 83 external products, the Application Set for the Framework-light application contains only 30 packages. The corresponding Framework-light release distribution includes 86 packages and 16 external products.

In the past two years a considerable development effort has been applied to correctly define the dependencies between the packages, package categories, and external software products in order to eliminate unnecessary connections. The Application Set and Build Set for the two major applications have been stabilized. The dependency discovery and checking procedures are still strictly required, as every development step potentially involves a change in the dependency pattern. Now, in this stable phase, we are changing over to a preventive strategy. Dependency checking is done for every software integration build so that any accidentally-introduced and unwanted dependencies are detected and cured at the early stage in the release integration process.

References

[1]http://cmsdoc.cern.ch/cms/outreach/html/, CMS Experiment
[2]http://cms.cern.ch/iCMS/jsp/page.jsp?mode=cms&action=url&urlkey=CMS_OFFLINE, CMS Offline Software
[3]D. Lange contribution to CHEP2009, Software Integration and Development Tools in CMS
[4]http://rpm.org RPMS Package Manager
[5]http://www.debian.org/doc/manuals/apt-howto, Advanced Package Manager
[6]https://cmstags.cern.ch/cgi-bin/CmsTC/CmsTCLogin, CMS Tag Collector
[7]http://indico.cern.ch/contributionDisplay.py?contribId=302&confId=3580, CMS packaging system

[8]https://twiki.cern.ch/twiki/bin/view/CMS/EventFilter, CMS Event Filter
[9]https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideFWLiteAnalysis, Framework-light Analysis Tutorial
[10]http://www.ihep.ac.cn/~chep01/paper/8-024.pdf, Ignominy: a tool for software dependency and metric analysis with examples from large HEP packages
[11]https://twiki.cern.ch/twiki/bin/view/CMS/HowToInstallONLINErelease, Online Releases installation guidelines
[12]https://twiki.cern.ch/twiki/bin/view/CMS/SWDevToolsWorkshopApr07, CMS Software Development Tools Workshop, agenda. Report on CMS Software Development Tools Workshop, Internal Note: CMS IN 2007/000
[13]https://twiki.cern.ch/twiki/bin/view/XdaqWiki, Xdaq, a platform for the development of distributed data acquisition system
[14]http://www.quattor.org, The quator administration tool suite
[15]http://root.cern.ch , ROOT