Available on CMS information server

CMS CR -2009/091

The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

12 May 2009

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 , Elizabeth Sexton-Kennedy

Abstract

The CMS Software project CMSSW embraces more than a thousand packages organized in subsystems for analysis, event display, reconstruction, simulation, detector description, data formats, framework, utilities and tools. The release integration process is highly automated by using tools developed or adopted by CMS. Packaging in rpm format is a built-in step in the software build process. For several well-defined applications it is highly desirable to have only a subset of the CMSSW full package bundle. For example, High Level Trigger algorithms that run on the Online farm, and need to be rebuilt in a special way, require no simulation, event display, or analysis packages. Physics analysis applications in Root environment require only a few core libraries and the description of CMS specific data formats. We present a model of CMS Partial Releases, used for preparation of the customized CMS software builds, including description of the tools used, the implementation, and how we deal with technical challenges, such as resolving dependencies and meeting special requirements for concrete applications in a highly automated fashion.

Presented at *CHEP 2009,21-27/03/2009,Prague,Czech Republic,15/05/2009*

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

UNI Experimentelle Kernphysik, Kaiserstr. 12, 76131 Karlsruhe, Germany

Email: Natalia.Ratnikova@cern.ch

Abstract. The CMS Software project CMSSW embraces more than a thousand packages organized in subsystems for analysis, event display, reconstruction, simulation, detector description, data formats, framework, utilities and tools. The release integration process is highly automated by using tools developed or adopted by CMS. Packaging in rpm format is a built-in step in the software build process. For several well-defined applications it is highly desirable to have only a subset of the CMSSW full package bundle. For example, High Level Trigger algorithms that run on the Online farm, and need to be rebuilt in a special way, require no simulation, event display, or analysis packages. Physics analysis applications in Root environment require only a few core libraries and the description of CMS specific data formats. We present a model of CMS Partial Releases, used for preparation of the customized CMS software builds, including description of the tools used, the implementation, and how we deal with technical challenges, such as resolving dependencies and meeting special requirements for concrete applications in a highly automated fashion.

1. Scope and Motivation

The CMS [1] software project CMSSW [2] includes over 1000 interdependent packages for data analysis, event display, reconstruction and simulation algorithms, detector description, data formats, framework, and utilities. It depends on about 100 of external software products.

Every external package, including configuration, development, and installation tools, is built and packaged for distribution. The whole CMSSW release is automatically configured, built and packaged into one distribution file. Prepared releases and externals are published for distribution and deployment on the GRID and on the user development machines.

Alongside with the main approach CMS has at least two important use cases, when only small subsets of packages and externals are needed:

- High Level Trigger (HLT) algorithms running online on the Event Filter farm [3].
- Light-weight Root based physics analysis tool Framework-light [4].

[1] To whom any correspondence should be addressed.

The model of Partial Releases allows building customized releases for such applications, while satisfying a number of special requirements.

The main technical challenges addressed by a model for the Partial Releases are the consistency with the base release by construction, and the automation of all steps of the procedure for optimal support at reduced maintenance cost.

2. Model and Implementation

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

- *Base Release* – is a given version of a standard CMS Software release, including the full set of CMSSW packages, and 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 set of packages of the base release, and external products, necessary to build all packages of the Application Set.
- *Partial Release* - is an independent software release of packages defined in the application Build Set, built and packaged using standard CMS release management tools.

As illustrated on Figure 1, the Base Release represents a large number of interdependent packages. The desired functionality of a particular application is provided by a well-defined subset of packages, we call it Application Set. Any package included in Application Set may depend on other packages beyond the Application Set. Dependencies occur at compilation time via included header files, at link time via symbols in libraries, and at run time via dynamically loaded plug-in modules. All these dependencies need to be satisfied for the package to be built and run successfully. Thus the partial release includes all packages and external products, required by the Application Set; we call it Build Set.

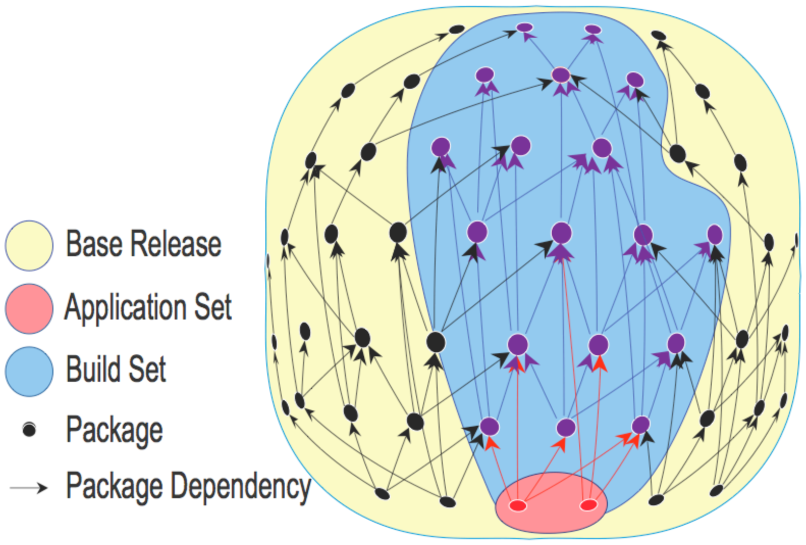

Figure 1 . Illustration of the partial release model.

The procedure of building the Partial Release starts from the request of the application manager, who defines the base CMSSW release, and corresponding Application Set. All the subsequent steps are performed automatically using standard tools developed by the CMS Collaboration.

The crucial point of finding the correct Build Set is discovery of dependencies between packages. We use the Ignominy tool [5] to extract and collect all dependency information for the base CMSSW release. Ignominy provides fine-grained information about direct dependencies of various types. For the Build Set we need to analyze package dependencies recursively, while discarding some types of dependencies, such as appearing from the package's local unit tests. This work is done by the BuildSet tool, which takes as input the Application Set and Ignominy output for the Base Release, and produces a list of packages and external products for the Build Set.

Once the Build Set is found, we start building the release. Specifications for release build and configuration are defined in the CMSDIST repository. The specifications for the Partial Release are constructed in such a way that the source code distribution of the corresponding Base Release is used automatically. All packages included into the Build Set are automatically selected and copied into the build area of the partial release. Thus, 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 packaging system [6]. Software build and configuration rules are defined in the shared area, so that Partial Release uses the same set of rules as the base CMSSW release does.

In this implementation the overall consistency of the code, external configuration, and build rules for the Partial and Base releases, is insured by construction.

3. Tools used for producing the Partial Release

- *IGNOMINY* is a powerful tool used to analyze software release source code and resulting build products, and to detect different types of dependencies between software packages.
- *BuildSet* tool allows to query Ignominy results, and to calculate dependencies recursively for a given package or a list of packages.
- *CMSDIST* is a repository of specification files for building distributions of CMSSW and required external products.
- *PKGTOOLS* provides a framework for building, packaging and uploading software distributions according to instructions in the CMSDIST specifications.
- *Tag Collector* [7] is the central interface to publish packages version tags and to manage the contents of CMSSW releases on a package level.
- *SCRAM* is the software configuration and release management tool used for managing CMSSW release internal configuration and build rules. It also provides CMS software development environment.

4. Examples of Applications

The concept of Partial Releases was driven by two major use cases: software distribution of the High Level Trigger algorithms for the online Event Filter Farm, and distribution of the light weight CMS software framework for physics analysis in Root environment.

4.1. Online release

The goal of the Online Release [8] is to provide consistent with Base Release environment for HLT online operation, while minimizing the amount of code to be deployed on the Event Filter Farm machines. It should also offer the possibility of producing patch releases forking from the mainstream CMSSW release sequence to retrofit necessary fixes while guaranteeing stable operation of the online HLT.

Special requirements [9] to build the Online Release are the following:

- Minimize online Build Set to improve robustness and stability.
- Use system compiler, hence different architecture name and completely separate set of distribution files.
- Use external packages available on the local system: online version of XDAQ software [10] and corresponding external products.

- Software distribution must be suitable for use with Quattor tool [11], which is used to manage software installations on the Event Filter farm, and has a number of constraints with respect to regular CMSSW installation method based on apt-get.
- Full rebuild may be required in case of major software upgrade on the Event Filter farm.

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 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 the 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 is distributed with the original "parent" release.

4.2. Framework-light release

The goal of the Framework-light application is to provide the possibility of convenient physics analysis of data in a regular CMS Event Data Model format in plain Root environment.

Special requirements:

- Minimal distribution set must only include Root and external packages required by Root, together with necessary Data Format packages.
- All dictionaries, necessary for analyzing Analysis Object Data (AOD) directly in Root, must be included.

Consequently, the Application Set for Framework-light release includes all packages containing objects which may appear in the Analysis Object Data (AOD), and corresponding analysis algorithms. A special effort was applied to eliminate any accidental dependencies that would bring additional unneeded packages into Framework-light distribution.

Small distribution size allows easy download and installation of Framework-light application onto the user's laptops. Minimal number of external dependencies and modest amount of code simplify porting to another platforms. It is already possible to run the Framework-light application on the MacOS.

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

5. Summary and Conclusions

The model of Partial releases allows building and distributing customized software builds for particular applications, in cases when the big size and complexity of the standard full CMSSW distribution becomes an issue. As an example, the Application Set for the Framework-light application currently contains 30 packages. The resulting Framework-light distribution includes 86 packages and 16 external products, while the corresponding full CMSSW Base Release includes 1114 packages and 83 external products.

In the past two years a successful development effort had been applied to clean up and straighten out 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 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] https://twiki.cern.ch/twiki/bin/view/CMS/EventFilter,
CMS Event Filter

[4] https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideFWLiteAnalysis,
FWLite Analysis Tutorial

[5] 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

[6] http://indico.cern.ch/contributionDisplay.py?contribId=302&confId=3580,
CMS packaging system

[7] https://cmstags.cern.ch/cgi-bin/CmsTC/CmsTCLogin,
CMS Tag Collector

[8] https://twiki.cern.ch/twiki/bin/view/CMS/HowToInstallONLINErelease,
Online Releases installation guidelines

[9] 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

[10] https://twiki.cern.ch/twiki/bin/view/XdaqWiki,
Xdaq, a platform for the development of distributed data acquisition system

[11] http://www.quattor.org,
The quator administration tool suite