

Executive Summary of Requirements for Software Product Building and Management

Revision 2.1.4

1 Introduction

This document is intended to serve as an executive summary to the document, “Requirements for Software Product Building and Management,” found in FNAL DOCDB as document no. 5380¹, hereafter referred to as, “the requirements document.” It is not intended to be a substitute for the detailed statements of all the requirements therein, but a synopsis of the overall aim of the larger document, and a specific call-out of some of the more impactful and essential requirements.

Use of terminology in this summary is fairly straightforward, but in any case terms are defined in the requirements document.

In the requirements document, requirements are categorized follows:

General: Requirements common to all categories.

Product management: Setup and use of coherent sets of related products.

Product development: Development, build and test of experiment-authored products.

Product integration: Parallel development of multiple experiment-authored products.

A build tool for release management: The building, installation and packaging of sets of related products.

In sum, these requirements describe all the software build, development, integration and management activities an experiment user, release manager, or developer would need.

The distinction between “behavior requirements,” “performance requirements,” and “constraints” made in the requirements document is not preserved here, and some requirements have been condensed together or omitted entirely for fluency and compactness. The requirements document remains the definitive statement, however.

There follows a section for each of the requirements categories listed above, followed by a final section of other considerations that, while relevant to the larger discussion, were not mentioned in the requirements document.

2 General Requirements

The platform requirements may be stated generally as:

¹<https://cd-docdbcert.fnal.gov/cgi-bin/cert/ShowDocument?docid=5380> as at publication time of this summary.

- The most current MacOS (Darwin) platform, and one previous.
- The most current Scientific Linux platform and one previous, but at minimum including the majority grid platform, and those (e.g. CentOS) compatible with same.
- The current Ubuntu LTS distribution and compatible derivatives.

All activities described by the requirements shall be possible while operating from a shell native to the supported platforms of either the “bash,” “tcsh,” or “zsh” flavors.

It shall be possible to carry out any of the activities described by the requirements using products from consistent releases while overriding one or more of those products with other builds of same, subject to ABI compatibility considerations.

3 Product Management

The Product Management System (PMS) makes products available for use and manages which products are active at any time in the current environment. This includes enabling access to executable programs and documentation, and any libraries, headers, and modules necessary for the development of other products.

The PMS shall be able to distinguish different versions of a product, and different “variants” of the same version. Variants of a product indicate different build options or environment (e.g. compiler, standard, versions of any dependencies, or product features).

The PMS shall be able to make active (set up) a single version and variant of a product, or a coherent collection of guaranteed-compatible products. All necessary dependencies shall also be activated, while ensuring compatibility with any already-active products and notifying the user in the case of consistency errors.

The PMS shall not require system privileges for its own installation, nor for the installation or activation of any product or set of products.

The PMS shall not require that it or any product be installed on a given filesystem or at a particular path in order for active products to operate successfully, and shall be able to deal both with pre-built binary products and products which have been built in-place.

The PMS shall be able to distinguish products built for different platforms, and to handle non-platform-dependent products, and to detect which products should be activated without requiring the user to specify the desired platform.

The PMS shall be able to handle the activation of > 100 products without violating system resource limits.

The PMS shall avoid requiring the installation of products already installed on the system under its management.

4 Product Development

The Product Development System (PDS) allows the development, building, testing, installation and packaging of a single product.

The PDS shall support all standard operations (“targets”):

build (a.k.a default or all): Running of the preprocessor (and saving of the output), code generation (including text substitution from build-time information), source to object code or bytecode compilation, production of shared libraries or precompiled modules, and production of executables and documentation.

test: The execution of configured tests: scripts, built executables or external programs, with specified arguments, and the evaluation of success or failure on exit code or the presence or absence of particular output. Tests shall be configurable to depend upon other tests, with the output of one or more forming the input of another, and output shall allow the ready identification of failed tests while allowing access to fully detailed test output as the user requires.

install: The installation of all final artifacts into an area distinct from the build area suitable for management by the PMS.

package: The packaging of the installation area into a product installation entity for installation and management by the PMS.

clean: The on-demand removal of products of the build and test stages.

help: The interrogation of the PDS to divine individual targets available in the current context.

The PDS shall allow out-of-source builds, and support reliably the parallel execution of as many build and test operations as compatible with specified interdependencies.

The PDS shall only execute those operations for which build products are not already present and valid, and shall additionally support command-line direction to build only specific targets or groups thereof.

The PDS shall support non-system compilers and cross-compilation.

The PDS shall support the command-line specification of configuration options (such as “debug,” “profile,” or “-DMY_FLAG=27”) for subsequent builds, and the option to show exactly what commands are executed for each build or test operation (“verbose” mode).

The PDS shall be extensible to support new languages, code generators, and rules to apply particular options to certain operations or groups of operations (e.g. compiler flags for sources used for plugins).

The PDS shall support the use of wildcards for sources of a particular build target.

The PDS shall support the identification of dynamic libraries with their versions in a way that allows programmatic interrogation, and additionally support the distinction between officially and privately built products.

The PDS shall support the use of external dependencies, including implicit use of dependencies at further levels of nesting, and especially the correct selection of appropriate variants of dependency for the current build configuration (interacting with the PMS as necessary).

5 Product Integration

The Product Integration System (PIS) allows the coordinated development, building, testing, installation and packaging of one or more products.

The PIS shall handle the setup of version- and variant-appropriate dependencies both between and external to the products to be built interacting as necessary with the PMS, including the detection of inconsistencies and missing intermediate dependencies required for a consistent build.

The PIS shall support the building of products using the PDS with only such additions to those products' build systems as remain compatible with standalone development of each product with the PDS.

The PIS shall not restrict the use of features of the PDS, including but not limited to as-needed rebuilding, single-target operations, and other partial rebuilds.

The PIS shall allow the simultaneous use of the same sources in distinct integrated builds, and shall place no constraints on the origins of product sources.

The PIS shall be configurable to automatically obtain required sources from SCM systems, including awareness, where appropriate, of branches and tags, and provide simplified interaction with known SCM servers.

The PIS shall be extensible to support new SCM systems and servers without having to alter the core system, and to support the creation of new product directory structures and insertion of same into a new or empty repository in any supported SCM system.

The PIS shall provide the ability to update the version number of a product, and to propagate this information to other products in the group to maintain consistency.

The PIS shall allow the user to identify the source, build, and installation areas used for a given integration build.

The PIS shall when appropriate execute build operations in parallel across products.

6 A Build Tool For Release Management

The Build Tool for Release Management (BT) is used for the building, installation and packaging of specified versions and variants of multiple products.

The BT shall allow the specification of the build instructions for a single product as a self-contained entity. The BT is responsible for locating the build instructions and/or pre-built installations of specified dependencies.

The build instructions for a product shall be required to specify only direct dependencies, and shall be able to do so in a way that is not duplicative of its interactions with the PMS.

The BT shall place no constraints on the choice of build system for any product, and additionally shall be able to support the build and use of products consisting only of dependencies.

The BT shall support the specification of build sets ("distributions") of not-necessarily-interdependent products, and the bulk building thereof.

The BT shall support the forced rebuild of one or more products in a distribution and the automatic rebuild of those products depending on them.

The BT shall support the build of distributions specifying multiple versions and/or variants of the same product, including the use of different compilers as appropriate.

The BT shall support the specification of build-only dependencies, including compiler choice, and the specification of platform- and compiler-independent dependencies.

7 Other Considerations

While not mentioned in the requirements document when it was written, the feature of a central repository for build instructions, binary products and distributions is one upon which experiments, and product and distribution providers, have come to rely.

CVMFS is mentioned in the requirements document as an existing deployment medium for products, but it is not explicitly mentioned as a requirement that this be the case.

Compatibility with one or more IDEs have been occasionally mentioned as a desire, but there has never been an explicit requirement for same.