Introduction

To allow comparison of packaging solutions, the WG should define a basic stack of packages to provide a test bed. The LCG stack is quite big, so the intent is to enumerate a common subset of this that could be used by a (small) experiment.

To help in collection and comparing requirements, the document is initially divided into two sections. The first lists known requirements for individual projects, the second merges these lists to define the test bed stack.

Packages used in Existing Stacks

The following sections list packages required by individual experiments/projects for both development and runtime use. These are direct dependencies including any version requirements plus enabled optional components. The lists are not necessarily in strict dependency order. Use of system packages is not considered just yet.

SuperNEMO (from Ben Morgan)

- GCC >= 4.9 (Linux only, macOS always uses Xcode)
- CMake >= 3.5
- Doxygen >= 1.8
- Boost >= 1.60
    - icu4c
- CAMP (being replaced with Ponder)
- Readline
- GSL >= 2
- CLHEP
- Xerces-C
- Geant4 >= 9.6.4
- Python >= 2.7
- ROOT >= 6
    - Python
    - GSL
    - OpenSSL
    - SQlite
- Qt5 (Core libraries only)
- Qt5Svg

FCC (from Benedikt Hegner)

- GCC >= 6.2
- ROOT >=6.10
- Geant4 >= 10

- Gaudi >= 29v1
- Python >= 2.7
- ACTS
- PODIO
- DD4hep
- Plus their dependencies

LHCb (from Ben Couturier)

- Gaudi >= v29r1
- Gcc > = 6.2
- ROOT >= 6.10.06 (c.f. LCG 91)
- BOOST => 1.64.0
- CLHEP >= 2.3.4.4
- GSL >= 2.1
- CppUnit
- HepPDT >= 2.06.01
- Python > = 2.7.13
- XercesC >= 3.1.13
- rangev3
- tbb
- Plus dependencies

Art (from Lynn Garren)

- GCC >= 6.3.0
- CMake >= 3.9.2
- Boost 1.65.1
- Sqlite 3.20.01.00
- Python 2.7.14
- TBB 2018
- CLHEP 2.3.4.5
- Cppunit 1.3.2
- ROOT 6.10.08
- Fftw 3.3.6.pl2
- Libxml2 2.9.5
- Xrootd 4.7.0
- Geant4 10.3.p01 and 10.2.p03
    - Not a direct art dependency
    - Optional link to Qt5

Miscellaneous

- Nvidia CUDA Toolkit 9.0

Wire-Cell

- Eigen3 >= 3.3.0
- FFTW3 >= 3.3.0
- Jsonnet >= 0.9.4
- JSONCPP >= 1.7.7

CMSSW FWLite 9.2.13 on macOS (Patrick Gartung)

- Apple Clang 8.1 or 9.0
- ROOT 6.11.02
    - Cfitsio 3.410
    - Fftw 3.3.6
    - Freetype 2.7.2
    - Libtiff 4.0.8
    - Libpng 1.6.29
- Boost 1.63.0
- Python 2.7.14
- Clang python bindings with libclang.dylib from homebrew llvm 4.0.1.
- CLHEP 2.3.4.2
- Expat 2.2.2
- HEPMC 2.06.09
- Libuuid 1.0.3
- Libxml2 2.9.4
- Gmake 4.2.1
- Scram 2.2.6
- Xerces-c 3.1.4
- Tbb 20161128oss
- Pcre 8.40
- Md5-cms 1.0.0
- Tinyxml-cms 2.5.3
- Fireworks-data (runtime)
- Vdt 0.3.9 (runtime)
- Xrootd 4.6.0 (runtime)

Combined Package List for the HSF Test Stack

The following stack results from merging the lists presented in the previous sections. It is listed in lowercase alphabetical order. When multiple versions of the same package have been listed, all version specifiers are listed for clarity. It does not yet fully resolve all dependencies

1. acts
2. boost >=1.60, 1.63.0, >=1.64.0, 1.65.1
3. camp (being replaced with ponder)
4. cfitsio 3.410
5. clang python bindings with libclang.dylib from homebrew llvm 4.0.1.
6. clhep 2.3.4.2, >=2.3.4.4, 2.3.4.5
7. cmake >=3.5, >=3.9.2
8. cppunit 1.3.2
9. dd4hep
10. doxygen >= 1.8
11. eigen3 >= 3.3.0
12. expat 2.2.2
13. fftw >= 3.3.0, 3.3.6, 3.3.6.pl2
14. fireworks-data
15. freetype 2.7.2
16. gaudi >= v29v1
17. gcc >= 4.9, >= 6.2, >= 6.3.0
18. geant4 >= 9.6.4, >= 10, 10.3.p01, 10.2.p03
19. gmake 4.2.1
20. gsl >= 2, >= 2.1
21. hepmc 2.06.09
22. heppdt >= 2.06.01
23. icu4c
24. jsoncpp >= 1.7.7
25. jsonnet >= 0.9.4
26. libtiff 4.0.8
27. libuuid 1.0.3
28. libxml2 2.9.4, 2.9.5
29. linpng 1.6.29
30. md5-cms 1.0.0
31. openssl
32. pcre 8.40
33. podio
34. python >= 2.7, >= 2.7.13, 2.7.14
35. qt5base
36. qt5svg
37. rangev3

38. readline
39. root >= 6, >= 6.10, 6.10.06, 6.10.08, 6.11.02
40. scram 2.2.6
41. sqlite 3.20.01.00
42. tbb 20161128oss, 2018
43. tinyxml-cms 2.5.3
44. vdt 0.3.9
45. xcode, 8.1, >=9.0 (macOS only)
46. xerces-c >= 3.1.13, 3.1.4
47. xrootd 4.6.0, 4.7.0

For a test stack, this is a non-trivial number of packages, especially when dependencies are considered. Given the requirements from experiments/projects it's possible to trim this slightly to a list of most commonly used packages. Newest versions should also be considered, though with consideration for OS level support.

Reuse of OS Packages

Package managers typically assume a "blank slate" system so that all dependencies can be resolved. Using a secondary package manager (e.g. Spack, Portage) on top of the system package manager (yum, apt etc) may result in the installation of packages that could be taken from the system install. Examples of packages that could be reused from the system are OpenSSL and X11. Fresh installs of packages through the secondary package manager may be best done when there is a requirement for

- A newer version than the system supplies
- A patched or custom install (e.g. additional components)

Whilst the Test Stack should enumerate all dependencies, a longer term aim should be to define a "Base System" of system packages that could be reused (plus, the tools available in package managers like Spack, Portage to enable this reuse).

On Linux, this will be highly distribution/version dependent, but for CentOS and Ubuntu systems the HEP_OSlibs meta-rpm/deb provides an example of how such a base system could be defined.

On macOS, the base system is pre-defined based on the OS/Xcode version.

Test Driving the Package Managers

Based on discussion in HSF Packaging Meeting #14, a minimal subset of packages was identified to exercise the package managers being looked at by the WG. To enable to WG,

and the broader community, to test drive the package managers, this stack has been combined with a basic checklist of tasks to create a template that each package manager can fill in to provide a “driving lesson”. To keep things simple to begin with, some simplifying assumptions are made:

1. Docker images are used for Linux testing. This is purely for consistency and reproducibility, and does not suggest that containers will be the only way to use a given package manager! It also helps to enumerate the OS packages that are always needed.
2. No use of CVMFS is assumed yet. This is so that test drivers can get a feel for building from source, installing from binary, writing packages for their own software, and the balance between reuse of OS packages vs “compile the world”. It does not imply that CVMFS will not be used later, but users will have to go through the packaging steps to have something to deploy to CVMFS! Smaller experiments may also not have access to CVMFS.
3. The test driver may have sudo access, but the steps requiring this should be minimized and ideally zero.

The following sections provide a draft template for each step, italic text is for things to be filled out for the specific package manager. It’s fine to just link to external documentation!

Test Driving the “X” Package Manager

Base Operating System Install

***Authors**: assume either a macOS High Sierra with Xcode 9, or a Docker Image based on centos:centos7 or ubuntu:xenial. In the Docker case, supply (a) Dockerfile(s) for each system, adding any additional system packages required, and finally create an “hsf” user with sudo privileges that the container will run with. There’s no requirement to build and host the images. Bonus points: Singularity! If macOS needs additional packages, document them below:*

Test driving “X” requires either a CentOS6/7, Ubuntu 16.04LTS, or macOS High Sierra system. For macOS, only the base system plus Xcode 9 from the App Store *[add any additional requirements here]* is required. For convenience and reproducibility, Docker images are available for Linux, and can be obtained and run as follows:

Add Docker pull/build/run instructions here

Optionally, you may use an existing Linux install, but you may encounter errors in subsequent steps if it is missing (or has incompatible) packages, or your environment has custom settings.

Installing “X”

To install “X”, *[Add instructions here, aim should be to get a base install with NO, or at least the most minimal set possible, installed. Include a set of test/sanity checks if the packages manager allows]*

Installing the Test Stack

The basic HSF Test Stack packages are as follows:

- Toolchain
  - GCC 6.4
    - With c, c++, fortran languages
  - Python 2.7.14
  - *[Add Any tools needed to build Stack packages]*
- Core Packages
  - Boost 1.65
  - ROOT 6.12.06
    - Including PyROOT, MathMore
  - GSL 2.4
  - Qt5 5.10 (Base system only)
  - Xerces-C 3.1.4
  - CLHEP *[Version to be compatible with Geant4]*
  - Geant4 10.3

[Add commands, instructions to install these packages from source and from binary if available]
*[**Optional:** Show other features, such as different C++ standards, optional components of ROOT]*

Adding a New Package

To add a new package to the stack, *[Add instructions here]*

Example: Homebrew

Not being considered, but here used as an example of filling out the template (Note that it doesn’t cover everything, nor the exact versions - that’s fine, and also for other package managers. We want to identify problem areas!

TODO (Ben)