Journal of Physics: Conference Series

Status and new developments of the Generator Services

Status and new developments of the Generator Services

B Hegner, P Mato, W Pokorski
PH Department, CERN, 1211 Geneva 23, Switzerland

A Karneyeu, M Kirsanov
Institute for Nuclear Research of Russian Academy of Sciences, Moscow, Russia

D Konstantinov, K Lugovskiy
Institute for High Energy Physics, Protvino, Russia

E-mail: `Witold.Pokorski@cern.ch`

Abstract. The Generator Services (GENSER) provide ready-to-use Monte Carlo generators, compiled on multiple platforms or ready to be compiled, for the LHC experiments. In this paper we discuss the recent developments in the build machinery, which allowed to fully automatize the installation process. The new system is based on and is integrated entirely with the "LCG external software" infrastructure, providing all the external packages needed by the LHC experiments.

1. Introduction

The Generator Services project [1] was started 10 years ago. Its purpose is to provide, mainly for the LHC community and experiments, the following components, necessary for the usage of Monte Carlo event generators:

- Installed, ready to use MC generator libraries on network file systems (e.g. AFS and CVMFS) for a number of computing platforms.
- Binary tarfiles for the distribution to the remote sites, for a number of platforms.
- Source tarfiles for building the generators from scratch. While for many generators these are the same as on the author's distribution site, for some generators, such as Pythia 6 and Herwig, GENSER provides a build system and an examples subdirectory, also with a build system.

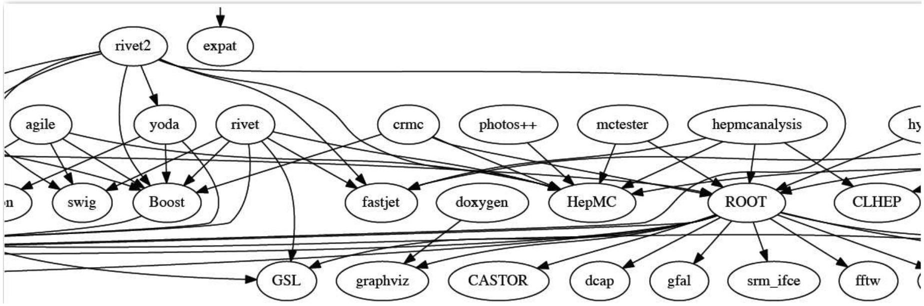

Figure 1. Part of the package dependencies diagram

Currently GENSER supports 34 generators and auxiliary packages. The most actively developed and used generators are **CRMC [2], EvtGen [3] [4], Herwig++ [5] [6], HYDJET [7] [8], HYDJET++ [9] [10], PHOTOS++ [11] [12], POWHEG-BOX [13] [14] [15] [16], PYTHIA 6 [17] [18], PYTHIA 8 [19] [20], SHERPA [21] [22], TAUOLA++ [23] [24].**

The most frequently used auxiliary packages are **AGILe [25], HepMC [26] [27], HepMCAnalysis [28] [29], LHAPDF [30] [31], MC-TESTER [32] [33], RIVET [34] [35].**

The list of supported computing platforms is decided in discussions with the representatives of the LHC experiments. Currently there are only Linux platforms in this list.

The active development of generators and auxiliary packages during the last 2 - 3 years caused the change of conditions for the Generators Services. The generators implemented in C++, such as pythia8, sherpa, herwig++, reached the production quality. They are now requested by the experiments more than the fortran ones. The new generators, unlike the fortran ones, usually have one or several dependencies on the LCG (LHC Computing Grid) external software packages, such as FastJet [36] [37], python, Boost (see Figure 1). An upgrade of one of these packages to a newer version necessitates the rebuild of all dependent generators. For this reason the structure of GENSER repositories was changed. Now it follows the LCG external software releases, coming out on the average every one - two months. In order to automatize the rebuilding of GENSER packages according to this scheme, the GENSER build machinery was completely rewritten and integrated with the LCG external software infrastructure. The new system, called **lcgcmake**, is based on CMake [38] , in particular on its ExternalProject [39] module.

2. GENSER build and install procedure with lcgcmake

As mentioned above, one of the purposes of **lcgcmake** is to build a new GENSER tree following a new release of LCG external software. Apart from this, from time to time there are requests from LHC experiments or generator authors to install a new generator version or to add a completely new generator to GENSER. In order to trace such requests a wide usage of JIRA [40] is adopted.

The stages, involved in the typical GENSER service request life-cycle are the following:

- **Receiving a new request**
  Typically the installation requests for new generators or new versions come from MC authors or the experiments. A JIRA ticket is immediately created.
- **Implementation**
  The code driving all the necessary steps to build one particular generator is typically 10 - 20 lines in the main steering file CMakeLists.txt (Appendix 1). Sometimes (a rare case) one

Site	Build Name	Update Files	Configure Error	Configure Warn	Build Error	Build Warn	Test Not Run	Test Fail	Test Pass	Build Time
macitois18.cern.ch	x86_64-mac108-gcc42-opt	0	0	0	0	0	0	0	32	22 hours ago
lxbsp0516.cern.ch	x86_64-slc5-gcc43-opt	0	0	0	4	1	0	18	38	Oct 07, 2013 - 12:12 CEST
sft-slc6-i686-ec-1	i686-slc6-gcc48-opt	0	0	0	0	0	0	2	54	Oct 07, 2013 - 12:01 CEST
sft-slc6-ec-4	x86_64-slc6-gcc48-opt	0	0	0	0	0	0	2	54	Oct 07, 2013 - 12:00 CEST

Site	Build Name	Update Files	Configure Error	Configure Warn	Build Error	Build Warn	Test Not Run	Test Fail	Test Pass	Build Time
macitois18.cern.ch	x86_64-mac108-gcc42-opt	0	0	0	0	0	0	0	34	Oct 07, 2013 - 12:10 CEST
lxbsp0516.cern.ch	x86_64-slc5-gcc43-opt	0	0	0	0	0	0	0	59	Oct 07, 2013 - 12:13 CEST
slc6-x86-64-ec3.cern.ch	x86_64-slc6-gcc47-opt	0	0	0	0	0	0	0	59	Oct 07, 2013 - 12:01 CEST
spi-slc6-ec-2	x86_64-slc6-gcc48-opt	0	0	0	0_{-6}	0_{-1}	0	0_{-36}	59^{+38}	Oct 07, 2013 - 12:00 CEST

Figure 2. CDash screenshot

or several auxiliary small files are needed. SVN [41] is used as a repository for configuration and steering files to ensure control and reproducibility.

- **Testing and integration**
  The entire software stack (MC generators and external software packages) is checked out automatically from SVN and built every day. The build status is monitored by CDash ([42], Figure 2). For every generator there is at least one integrity test: an executable is built and run. These are short tests, taking not more than a few minutes, only pass/fail status is checked. The integrity tests are run automatically after the generator build using CTest [43]
- **Signing off and installation**
  If the results of the test builds and integrity tests described above are OK, the changes necessary to satisfy the request are approved for the release.

3. Some details of the implementation

The CMake ExternalProject module allows to define custom targets to drive download, update, patch, configure, build, install and test any external package. The code that defines these steps is a part of lcgcmake and is rather stable now.

At the initialization lcgcmake generates Makefile, with which the actual build is performed.

A simple make -jN is sufficient to build all generators and all the externals (about 120 packages) in about 2 hours. The ElectricCommander [44] procedures automate the whole building and installation process.

It was decided that the author's source tarfiles used to build packages should be taken from the GENSER intermediate repository, where they are put once and forever. This makes GENSER more self-contained, ensuring reproducibility of the build results.

The executables for the integrity tests mentioned above are built with CMake build systems. In these systems the CMake command *find_package* is widely used. For the standard packages like Boost there are corresponding modules built in CMake. For other, less standard packages like HepMC, the corresponding modules are collected in the package **cmaketools** installed as a part of LCG external software. This package can be useful for any project with a CMake build system that use generators.

4. Conclusions

The Generator Services are widely used in production by the LHC experiments. However, an intensive development of GENSER continues in order to make it more reliable, convenient and flexible when used on a variety of commonly used computing platforms. The most recent major development is the migration to CMake described in this paper. It allowed to make the installation of the whole new GENSER tree corresponding to a new LCG external software release fully automatic and steerable from ElectricCommander, essentially reducing it to a single click of button. This development was performed, as usual, without interruption of the main services for the LHC experiments.

5. References

[1] Kirsanov M, Ribon A, Zenin O 2008 Development, validation and maintenance of Monte Carlo event generators and generator services in the LHC era *PoS(ACAT08)* 114
[2] CRMC website: `http://www.auger.de/~rulrich/crmc.html`
[3] Lange D J 2001 The EvtGen particle decay simulation package *Nucl. Instrum. Methods* A **462** 152
[4] EvtGen website: `http://evtgen.warwick.ac.uk/`
[5] Bahr M et al. 2008 Herwig++ physics and manual *Eur. Phys. J.* C **58** 639-707
[6] Herwig++ website: `https://herwig.hepforge.org/`
[7] Lokhtin I, Snigirev A 2006 *Eur. Phys. J.* C **46** 211
[8] HYDJET website: `http://lokhtin.web.cern.ch/lokhtin/hydro/hydjet.html`
[9] Lokhtin I et al. 2009 Heavy ion event generator HYDJET++ *Comput. Phys. Commun.* **180** 779-799
[10] HYDJET++ website: `http://lokhtin.web.cern.ch/lokhtin/hydjet++/`
[11] Davidson N, Przedzinski T, Was Z 2010 PHOTOS Interface in C++ technical and physics documentation arXiv:1011.0937 [hep-ph]
[12] PHOTOS++ website: `http://photospp.web.cern.ch/photospp/`
[13] Nason P 2004 *JHEP* **0411** 040
[14] Frixione S, Nason P, Oleari C 2007 *JHEP* **0711** 070
[15] Alioli S, Nason P, Oleari C, Re E 2010 *JHEP* **1006** 043
[16] POWHEG-BOX website: `http://powhegbox.mib.infn.it/`
[17] Sjostrand T, Mrenna S, Skands P 2006 PYTHIA 6.4 physics and manual *JHEP* **0605** 026
[18] PYTHIA 6 website: `http://pythia6.hepforge.org/`
[19] Sjostrand T, Mrenna S, Skands P 2008 A brief introduction to PYTHIA 8.1 *Comput. Phys. Commun.* **178** 852-867
[20] PYTHIA 8 webiste: `http://home.thep.lu.se/~torbjorn/Pythia.html`
[21] Gleisberg T et al. 2009 Event generation with SHERPA 1.1 *JHEP* **0902** 007
[22] SHERPA website: `http://www.sherpa-mc.de/`
[23] Davidson N, Nanava G, Przedzinski T, Richter-Was E, Was Z Universal interface of TAUOLA technical and physics documentation arXiv:1002.0543
[24] TAUOLA++ website: `http://tauolapp.web.cern.ch/tauolapp/`
[25] AGILe website: `http://agile.hepforge.org/`

[26] Dobbs M and Hansen J B 2001 The HepMC C++ Monte Carlo event record for High Energy Physics *Comput. Phys. Commun.* **134** 41-46
[27] HepMC website `http://lcgapp.cern.ch/project/simu/HepMC/`
[28] Ay C, Johnert S, Katzy J, Qin Z 2010 HepMCAnalyser: A tool for Monte Carlo generator validation *J. Phys. Conf. Ser.* **219** 032029
[29] HepMCAnalysis website: `http://hepmcanalysistool.desy.de/`
[30] Whalley M, Bourilkov D, Group R 2005 The Les Houches Accord PDFs (LHAPDF) and Lhaglue arXiv:hep-ph/0508110
[31] LHAPDF website: `http://lhapdf.hepforge.org/`
[32] Davidson N, Golonka P, Przedzinski P, Was Z 2011 MC-TESTER v. 1.23: a universal tool for comparisons of Monte Carlo predictions for particle decays in high energy physics *Comput. Phys. Commun.* **182** 779
[33] MC-TESTER website: `http://mc-tester.web.cern.ch/MC-TESTER/`
[34] Buckley A et al. 2013 Rivet user manual *Comput. Phys. Commun.* **184** 2803
[35] RIVET website: `http://rivet.hepforge.org/`
[36] Cacciari M, Salam G P and Soyez G 2012 FastJet user manual *Eur. Phys. J.* C **72** 1896
[37] FastJet website: `http://fastjet.fr`
[38] `http://www.cmake.org/`
[39] `http://www.kitware.com/media/html/BuildingExternalProjectsWithCMake2.8.html`
[40] `https://confluence.atlassian.com/display/JIRA/JIRA+Documentation`
[41] `http://subversion.apache.org/`
[42] `http://www.cdash.org/`
[43] CTest testing tool
`http://cmake.org/Wiki/CMake/Testing_With_CTest`,
`http://www.cmake.org/cmake/help/v2.8.8/ctest.html`
[44] `http://www.electric-cloud.com/products/electriccommander.php`

6. Appendix 1

Fragment of the main steering file in the section of generators (CMakeLists.txt). The first part defines the details of the building and installation steps for one particular generator. The second part defines the corresponding integrity test.

```
#---Sherpa-----------------------------------------------------------------------------------------
LCGPackage_Add(
  sherpa
  URL http://cern.ch/service-spi/external/tarFiles/MCGeneratorsTarFiles/SHERPA-MC-<sherpa_<NATIVE_VERSION>_author>.tar.gz
  CONFIGURE_COMMAND ./configure --prefix=<INSTALL_DIR>
     --enable-shared --enable-static --enable-binreloc --enable-analysis
     --enable-hepevtsize=<sherpa_<NATIVE_VERSION>_hepevt>
     --enable-multithread
     --enable-lhapdf=${lhapdf_home}
     --enable-hepmc2=${HepMC_home}
     --enable-rivet=${rivet_home} "CFLAGS=-O2 -g0" "CXXFLAGS=-O2 -g0" "FFLAGS=-O2 -g0"
     --enable-openloops=.
  INSTALL_COMMAND make install
  BUILD_IN_SOURCE 1
  DEPENDS lhapdf HepMC rivet
)

# $PWD is needed because Sherpa uses this variable to know current directory
LCG_add_test(sherpa_orig_test1
  PRE_COMMAND ${CMAKE_COMMAND} -E make_directory sherpa/tests
  TEST_COMMAND ${CMAKE_COMMAND} -E chdir sherpa/tests ${sherpa_home}/bin/Sherpa -j20
                                   -f ${CMAKE_SOURCE_DIR}/generators/sherpa/tests/LHC_Z.dat
  POST_COMMAND ${CMAKE_COMMAND} -E remove_directory sherpa/tests
  ENVIRONMENT
    ${library_path}=${sherpa_home}/lib:${HepMC_home}/lib:${lhapdf_home}/lib:${fastjet_home}/lib:
                    ${GSL_home}/lib:$ENV{${library_path}}
    PWD=.
)
```