SANDIA REPORT

SAND2018-4421
Unlimited Release
Printed April 2018

Installation and Testing Instructions for the Sandia Automatic Report Generator (ARG)

Philippe P. Pébaÿ, Robert L. Clay

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology and Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2018-4421
Unlimited Release
Printed April 2018

Installation and Testing Instructions for the Sandia Automatic Report Generator (ARG)

Philippe P. Pébaÿ
NexGen Analytics
30 N. Gould St, Suite 5912
Sheridan, WY 82801, U.S.A.
philippe.pebay@ng-analytics.com

Robert L. Clay
Sandia National Laboratories
P.O. Box 969
Livermore, CA 94551, U.S.A.
rlclay@sandia.gov

Abstract

In this report, we provide detailed and reproducible installation instructions of the Automatic Report Generator (ARG), for both Linux and macOS target platforms.

Acknowledgments

The authors would like to thank Jerry Friesen (Sandia) for having installed and provided the Linux test platform used in the context of this work. We also thank our Sandia colleagues who have been involved in this project, in alphabetical order: Ernest "Foss" Friedman-Hill, Ed Hoffman, Tim Kostka, George Orient, and Nathan Spencer.

Contents

This page intentionally left blank

1 Generalities

The goal of this report is to provide detailed and reproducible installation instructions of the Automatic Report Generator (ARG), currently being developed at NexGen Analytics with funding and support from Sandia National Laboratories as part of the ASC Integrated Workflow Project (IWP), cf. [iwf18].

Furthermore, this document distinguishes between the two platforms on which the ARG is currently supported: namely, Linux (specifically in its CentOS distribution) and macOS.

Please note that Windows is not currently supported but that future plans include it as a target platform: as a result the current instructions will be updated in this regard as soon as possible.

1.1 Obtaining the ARG

In order to obtain the ARG, you will firstly need to be granted access by one of the authors of this report: please contact either of them in this regard.

Once access has been granted, the ARG can be either downloaded via the web interface of GitLab at `https://gitlab.com/ng-analytics-admin/IWF`, or interactively using Git in the command line of a terminal as follows:

```
git clone git@gitlab.com:ng-analytics-admin/IWF.git
```

Please note that when working from behind a firewall, the `https_proxy` variable must be set (ideally, added to the default user shell profile) in order to allow for the cloning command to go through said firewall.

As a result of either method, you will obtain (after unpacking the downloaded tarball in the former approach) a directory called `IWF`, directly under which a sub-directory called `ARG` is located. Of particular interest in the context of this report are the various cases containted in the `tests` sub-directories.

1.2 Dependencies

The ARG is written in Python, version 2, and is known to work with versions 2.7.10 and above thereof. It is therefore required that the target platform provide such a version of Python as no pre-compiled binaries of the ARG are currently distributed.

In addition, the ARG currently only features a LaTeX backend (although alternatives will be offered in the future). It is therefore required that a reasonably complete installation of LaTeX be available on the target system. Please note that we have successfully tested the ARG

with both Tex Live and XeTeX distributions, although only the former will be considered in the current report.

Aside from these the following dependencies are either required or optional:

1. Required:
   (a) SEACAS, which can be (without prior authentication):
      - either downloaded at `https://github.com/gsjaardema/seacas`
      - or fetched interactively with Git as follows:

      ```
      git clone https://github.com/gsjaardema/seacas.git
      ```

   (b) Exomerge, which can be (without prior authentication):
      - either downloaded at `https://github.com/gsjaardema/seacas`
      - or fetched interactively with Git as follows:

      ```
      git clone https://github.com/timkostka/exomerge.git
      ```

   (c) The following Python packages:
      i. `pyyaml`
      ii. `pylatex`
2. Optional but mandatory if logging remotely on the system where the ARG is installed:
   (a) OSMesa, in order to allow for off-screen GL rendering, which may be built from source, as indicated at `https://www.mesa3d.org/osmesa.html`, or simply obtained as pre-compiled package as will be discussed in the platform-specific chapter §2.5. This is the best method in our opinion in order to execute the visualization part of the ARG when connected remotely on a system, hereby alleviating all problems related to X forwarding. In contratst, if the ARG is going to be executed locally (e.g., on a laptop), then it is better to use the hardware rendering capabilities and therefore to not use OSMesa.
3. Optional:
   (a) The Visualization Toolkit (VTK, cf. [VTK10]), if it is desired that the ARG produce 3-dimensional visualizations of data sets. In the context of the ARG, it is recommended to build VTK from source (a process that we explain in the subsequent chapters on a per-platform basis), using a recent stable version, but not too recent in order to avoid problems due to the requirement of using C++11 with versions 8 and above. We therefore settled with version 7.1, and the remainder of this report is written under this assumption. As mentioned above, we recommend that the VTK rendering libraries link against OSMesa and **not** the system GL ones when deploying the ARG on a remote system.

(b) Octave or MATLAB, in order to allow for the execution of N. SPENCER's log file post-processing toolset. Please note that all work reported here has been done with the Octave; although the post-processing routines were originally written specifically for MATLAB, they have proven to be fully portable to the free software alternative.

(c) The following Python packages:

   i. `matplotlib`, if it is desired that the 2-dimensional plotting capabilities of the ARG be made available.

   ii. `oct2py`, in order to allow for the Octave to Python bridging between the ARG and the aforementioned log file post-processing routines.

   iii. `scipy`, which at least with some distributions appears to be a dependency of `oct2py` not accounted for by Pip.

Regarding Python packages, we recommend you install those system-wide, using the Pip package installer utility, as follows (this requires administrator privileges):

```
sudo pip install <package-name>
```

2 Linux Specifics

We firstly acknowledge here are that our Linux deployment was made using a CentOS distribution, version 7.4. The ARG deployment should nonetheless be similar, *modulo* minor adjustments, with other mainstream flavors of Linux.

All recommended installation procedures and settings required administator privileges, so they can be made system-wide.

2.1 Environment Settings

We are working under the assumption that all system-wide installations performed to meet the general requirements of §1 were built and/or installed as shared libraries, within the `/usr/local` sub-directory tree.

It is therefore necessary, in order that these libraries be found by the linker, that the `LD_LIBRARY_PATH` environment variable be set accordingly, either as follows:

```
export LD_LIBRARY_PATH="/usr/local/lib:${LD_LIBRARY_PATH}"
```

or, better yet, that the line above be included in the default user shell profile.

Similarly, the `PYTHONPATH` environment variable must be amended to so the locally-installed packages be found:

```
export PYTHONPATH="/usr/local/lib:${PYTHONPATH}"
```

Note that the `/usr/local/lib/python2.7/site-packages:` sub-directory must be added in second position to the above when VTK was also installed (cf. details in §2.5).

2.2 Python Dependencies

The easiest way to obtain the Pip package installer for Python, version 2, is to use the Yum package manager for Linux as follows:

```
sudo yum install python2-pip
```

In addition, after Exomerge has been obtained as indicated in §1.2, its Python library must be manually copied to where it can be found globally, as follows (assuming one is located in the top-level directory of the Exomerge source tree):

```
sudo cp exomerge.py /usr/local
```

2.3 SEACAS

A pre-requisite to building SEACAS from source (a requirement as indicated in §1.2) is to have CMake installed on the target system, which can be done simply as follows:

```
sudo port install cmake
```

We have found that building and installing SEACAS was fairly straightforward on a Linux/CentOS system, when following the detailed instructions provided in the `README.md` file that can be found at the top level of the source tree.

In particular, the `CMAKE_INSTALL_PREFIX` variable of CMake should be set to `/usr/local` by default which, after having executed

```
cd build
make
sudo make install
```

will result in installing the SEACAS libraries in `/usr/local/lib`, most notably the `exodus.py` Python module which is required by Exomerge.

2.4 LaTeX

The simplest way to go about fully installing Tex Live on your target Linux (CentOS-style) platforms is to issue the following command:

```
sudo yum install texlive-*
```

This command will produce a non-minimal installation, because the Tex Live distribution on Linux contains hundreds of packages, many of which are not needed by the ARG.

We have discovered that, at the time of writing, the Tex Live distribution of LaTeXthat is available on CentOS does not appear to provide the `spverbatim` package, a problem that we have resolved by installing manually the missing package, system-wide, by getting it from the CTAN archive; please refer to `https://ctan.org/pkg/spverbatim` for details.

2.5 OSMesa and VTK

In order to allow for all GL-based rendering in a software, off-screen fashion, using OSMesa and all necessary dependencies, which we installed on the target platform as follows:

```
sudo yum install mesa
sudo yum install mesa-libOSMesa
sudo yum install glut
sudo yum install glut-dev
sudo yum install glut3
sudo yum install freeglut-devel
sudo yum install libXt
sudo yum install libXt-devel
sudo yum install mesa-common
sudo yum install mesa-common-devel
sudo yum install mesa-libOSMesa-devel
sudo yum install freeglut-devel
```

VTK itself can be built relatively easily, as fully explained in the download and build instructions at `https://www.vtk.org/Wiki/VTK/Configure_and_Build`, and miust also be installed system-wide. Because all general instructions can already be found in the aforementioned online instructions, we only mention here what is specific to the ARG context; specifically, here are the CMake variables that must be adjusted, either manually in the top-level `CMakeCache.txt` file of the VTK build directory, or through the `ccmake` user interface:

```
BUILD_TESTING:BOOL=OFF
CMAKE_BUILD_TYPE:STRING=Release
CMAKE_INSTALL_PREFIX:PATH=/usr/local
VTK_ENABLE_VTKPYTHON:BOOL=ON
VTK_DEFAULT_RENDER_WINDOW_OFFSCREEN:BOOL=OFF
VTK_OPENGL_HAS_OSMESA:BOOL=OFF
OSMESA_INCLUDE_DIR:PATH=/usr/include
OSMESA_LIBRARY:FILEPATH=/usr/lib64/libOSMesa.so
```

Please note that you might have to manually adjust the paths values automatically detected by CMake for the Python and OSMesa libraries and include directories, but that should not be necessary if the global environment variables where priorly set as recommended in §2.1.

Once the VTK configuration and `Makefile` generation process have been performed with CMake, simply execute the following:

```
make
sudo make install
```

This will result in installing all requested VTK libraries under `/usr/local`.

3 macOS Specifics

We have been able to deploy the ARG following the instructions detailed here on both 10.12 (Sierra) and 10.13 (High Sierra) targets.

3.1 Environment Settings

In contrast with the Linux case, we are now making the assumption that all system-wide installations performed to meet the general requirements of §1 are built and/or installed as shared libraries under `/opt/local` (in contrast with the `/usr/local` of the linux case).

It is therefore necessary, in order that these libraries be found by the linker, that the `LD_LIBRARY_PATH` environment variable be set accordingly, ideally by including the following line in the default user shell profile:

```
export LD_LIBRARY_PATH="/opt/local/lib:${LD_LIBRARY_PATH}"
```

Similarly, the `PYTHONPATH` environment variable must be amended to so the locally-installed packages be found:

```
export PYTHONPATH="/opt/local/lib:${PYTHONPATH}"
```

Note that the `/opt/local/lib/python2.7/site-packages:` sub-directory must be added in second position to the above when VTK was also installed (cf. details in §3.5.

Finally, if the MacPorts version of Pip was installed (as we recommend), then the following value:

`/opt/local/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-package`

must also be added to the `PYTHONPATH`.

3.2 Python Dependencies

We have obtained all required Python dependencies for the ARG on macOS from the MacPorts project. Altough alternatives exist (e.g., Homebrew), in our experience MacPorts has proven to be both reliable and consistent across different maOS versions and is therefore the approach we recommend here. Please refer to `http://www.macports.org` for details as to how MacPorts can be obtained and installed on your macOS system.

Using MacPorts, the required version of Python can be easily installed as follows:

```
sudo port install python27
```

Once this is done, the appropriate version of Pip can be obtained and selected as follows:

```
sudo port install py-pip sudo port select --set pip pip27
```

In addition, after Exomerge has been obtained as indicated in §1.2, its Python library must be manually copied to where it can be found globally, as follows (assuming one is located in the top-level directory of the Exomerge source tree):

```
sudo cp exomerge.py /opt/local
```

3.3 SEACAS

A pre-requisite to building SEACAS from source (a requirement as indicated in §1.2) is to have CMake installed on the target system, which can be done simply as follows:

```
sudo port install cmake
```

We have found that building and installing SEACAS was a little more complicated on macOS, as compared to the Linux case (cf. §2.3), because the instructions provided in the `README.md` file that can be found at the top level of the source tree do not explain in detail how the third-pary libraries installer `install-tpl.sh` must be amended in order to force an installation under `/opt/local` instead of the Linux-style `/usr/local`.

The solution we have found has been to modify all `CMAKE_INSTALL_PREFIX` variables inside the `CMakeCache.txt` files initially generated for each of the third-party libraries, then to re-configure and re-create Makefiles for all of these with a separate pass of CMake, and finally to build/install these again, individually, with

```
make
sudo make install
```

Only after the third-party libraries are installed under `/opt/local`, can SEACAS per se be built and installed, as follows:

```
cd build
make
sudo make install
```

wich will result in installing the SEACAS libraries in `/opt/local/lib`, most notably the `exodus.py` Python module which is required by Exomerge.

3.4 LaTeX

We have also used the MacPorts route in order to install LaTeX on our macOS platforms, specifically in its Tex Live distribution.

In this context, we have found the following combination of ports to be sufficient in order to cover all of ARG needs (and those might not all even be strictly necessary):

```
sudo port install texlive-basic
sudo port install texlive-bin
sudo port install texlive-common
sudo port install texlive-fonts-recommended
sudo port install texlive-latex
sudo port install texlive-latex-extra
sudo port install texlive-latex-recommended
sudo port install texlive-math-science
sudo port install texlive-pictures
```

3.5 VTK

We first remark that for our macOS installations, which were meant for laptops where the ARG was to be used locally, we have not used the off-screen rendering facilities of VTK and have therefore not tested it with OSMesa there. However if it is needed to launch the ARG remotely on a macOS system, then the instructions of §2.5 for OSMesa should remain globally valid.

VTK itself can be built relatively easily, as fully explained in the download and build instructions at `https://www.vtk.org/Wiki/VTK/Configure_and_Build`, and miust also be installed system-wide. Because all general instructions can already be found in the aforementioned online instructions, we only mention here what is specific to the ARG context; specifically, here are the CMake variables that must be adjusted, either manually in the top-level `CMakeCache.txt` file of the VTK build directory, or through the `ccmake` user interface:

```
BUILD_TESTING:BOOL=OFF
CMAKE_BUILD_TYPE:STRING=Release
CMAKE_INSTALL_PREFIX:PATH=/opt/local
VTK_ENABLE_VTKPYTHON:BOOL=ON
VTK_DEFAULT_RENDER_WINDOW_OFFSCREEN:BOOL=OFF
VTK_OPENGL_HAS_OSMESA:BOOL=OFF
```

Please note that you might have to manually adjust the paths values automatically detected by CMake for the Python libraries and include directory, but that should not be necessary if the global environment variables where priorly set as recommended in §3.1.

Once the VTK configuration and `Makefile` generation process have been performed with CMake, simply execute the following:

```
make
sudo make install
```

This will result in installing all requested VTK libraries under `/opt/local`.

4 Testing and Support

In order to test your deployment of the ARG, using a shell terminal, please `cd` into the `tests` directory of the ARG source tree and simply run

```
sudo port install cmake
```

You will then see several dozens of lines of text output, detailing exactly the operations that the ARG, in its various components, is performing.

As a final result, you shall obtain the following six SAND reports in PDF format:

```
SAND2018-DropPackage.pdf
SAND2018-ModalPackage.pdf
SAND2018-ModalPackage_2.pdf
SAND2018-PLOTS.pdf
SAND2018-can.pdf
SAND2018-disk.pdf
```

In case you encounter any difficulty in this process, or the SAND reports are not or incorrectly generated, please provide detailed feedback to the authors of this document.

References

[iwf18] IWF Confluence Wiki, 2018.

[VTK10] *The VTK User's Guide, version 5.4.* Kitware, Inc., 2010.

DISTRIBUTION:

2	MS 9018	Central Technical Files, 8944
1	MS 0899	Technical Library, 9536


Sandia National Laboratories