Building SheafSystemTest™ on Windows

# Copyright Notice

© 2016 Limit Point Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

# Platform

SheafSystemTest, the test suite for the SheafSystem™ libraries, is supported for Windows 10 using Visual Studio 2015.

# Software Prerequisites

* CMake 3.5.0. Cross platform build tool; down load from www.cmake.org.
* Visual Studio 2015. C++ compiler; available from Microsoft.
* 7-zip 9.20 or newer. File archiving utility; down load from www.7-zip.org.
* SheafSystem, installed libraries or source tree.

Binary distributions with Windows installers are available for CMake and 7-zip; just use the default installation procedure.

# Building and running SheafSystemTest

Extract the package in a directory of your choice.

We'll assume you've down loaded the source as a zip file, SheafSystemTest-<version>.zip. (For instance, the SheafSystemTest Github page provides a link to download SheafSystem-master.zip). In Windows Explorer, with 7-zip installed, just navigate to the folder containing the down load and right-click on the file and select one of the extraction options to extract into a location of your choice. The package will extract into <your choice>\SheafSystemTest-<version>. From here on we'll refer to that location as <sheaf\_system\_test\_source>

Configure with CMake

Start the CMake application. In the "where is the source code" box enter or browse to <sheaf\_system\_test\_source>. In the "where to build the binaries" box enter <sheaf\_system\_test\_source>/build. Click on the configure button, then click the yes button to create the build directory and click the finish button to accept the default generator (Visual Studio). Configuring will start, it may take a while. Typically, several messages will be displayed about not being able to find some prerequisites.

There are two check boxes in the upper right part of the display. The "Advanced" check box toggles the display between "basic" and "advanced" mode. In basic mode the display shows only the variables you need to set to configure the system. In advanced mode, the display shows a large number of variables detailing the configuration process. Toggle the display to basic mode.

The "Grouped" check box toggles the display between "ungrouped" and "grouped" modes. In "ungrouped" mode the variables are listed in alphabetical order; in "grouped" mode, the variables are organized into an outline-like display. Choose whichever mode you find most appealing, but if in "grouped" mode, expand the headings so that all variables are visible.

There are two groups of variables you need to review, and perhaps set, to configure the SheafSystem: the SHEAFSYSTEM\_ variables, and the PREREQ\_ variables.

Set the SHEAFSYSTEM\_ variables.

The SHEAFSYSTEM\_ variables control options for building the system. The direct entry and command line entry methods apply to these variables.

There is only one SHEAFSYSTEM\_ variable:

SHEAFSYSTEM\_TEST\_JOBS (type STRING): the number of test jobs CTest will execute concurrently. Concurrent tests make the testing take less elapsed time but can make the results harder to read.

Set the PREREQ\_ variables.

The PREREQ\_ variables control the search for the prerequisites. There are three methods for setting these variables: direct entry, command line entry, and environment variable entry. Direct entry and command line entry are as described above. To use environment variable entry, set an environment variable of the same name to the desired value before invoking ccmake. Note that no matter which of these methods is used, it is important to set the value correctly. Incorrect values may produce unpredictable and hard to interpret results. In this case, it is often best to just delete the build directory and try again from Step 5!

When the prerequisites are all found successfully, CMake will write the file set\_prereq\_vars.bat into the build directory. This file is a batch script for setting environment variables for all the PREREQ\_ variables. It can be used on subsequent builds to simplify setting the PREREQ\_ variables - just run the script in a command window before running CMake.

There is one PREREQ variable:

PREREQ\_SHEAFSYSTEM\_CONFIG\_DIR (type PATH): the absolute path to the directory containing the file SheafSystemConfig.cmake. For instance, <installation tree root>/cmake or <source tree root>/build/cmake.

Click the configure button again. If it completes without error, click the generate button. Otherwise, correct the variables as needed and click configure again. When you've successfully generated, you're done, exit CMake.

Build and run

In the Windows Explorer navigate to <sheaf\_system\_test\_source>/build. Double click on SheafSystemTest.sln to start Visual Studio. In Visual Studio select Debug\_contracts in the Solution Configurations box. In the Solution Explorer pane, right click on the "check" target and select Build. Visual Studio will build and execute the test suite for the Debug\_contracts configuration of the SheafSystem libraries. The test suite contains several hundred test executables. Repeat for the RelWithDebInfo\_no\_contracts configuration. The Debug\_contracts tests take quite a while; RelWithDebInfo will run much faster.