

**Mississippi State University (MSU)
High Performance Computing Collaboratory (HPC²)
Center for Advanced Vehicular Systems (CAVS)**

Developer Installation and Setup Instructions

1. Download the archive file for each SimSys Package that is to be installed.

Go to the SimSys Software Forum and login using your username and password at <http://www.simcenter.msstate.edu/software/forum/login.php>. Then select the appropriate package link from those at the top of the page. For each package that is available on the forum there are three sub-folders named *archive* (for older versions), *release* (for the current release), and *special* (for special files if available and as needed). Each folder will contain SimSys Package archive files that are self-contained and include all files for a working system. Also, each folder contains descriptive documents such as this one, including README.pdf, LICENSE.pdf, README_packages.pdf.

2. Create a SimSys installation directory.

Create an installation directory for all SimSys Packages, such as “*simsys*”. Future upgrades can be installed over existing files or multiple directories can be used for each release/upgrade, e.g. *simsys_release*, *simsys_sept_rel*, *simsys_july_rel*, etc. If multiple people will be accessing the files then a global location would be best. The installation directory is static after the files are installed. There is no need for individual users to write within the installation directory.

3. Install the SimSys Package files.

The following Linux/MacOSX terminal commands will uncompress and untar the SimSys Package files.

```
cd simsys (or other installation directory)
tar -zxvpf location_of_package_tar_files/package_archive_file
```

Repeat the command for each package file. Alternatively, you can do the equivalent using Linux or MacOSX standard GUI. On WINDOWS systems use the GUI to unzip the files. After installing the files the following directories should exist.

<i>bin/</i>	user <i>sh</i> -shell script files and executables (for one <i>arch</i> specific system only)
<i>bin/dlib</i>	dynamic/shared libraries (if <i>AFLR4</i> is installed)
<i>interface/sm</i>	GUI definitions (if <i>SolidMesh</i> is installed)
<i>doc/system</i>	html-based documentation files for the overall <i>SimSys</i> system
<i>doc/aflr2</i>	html-based documentation files for <i>AFLR2</i> (if installed)
<i>doc/aflr2c</i>	html-based documentation files for <i>AFLR2C</i> (if installed)
<i>doc/aflr3</i>	html-based documentation files for <i>AFLR3</i> (if installed)
<i>doc/aflr4</i>	html-based documentation files for <i>AFLR4</i> (if installed)
<i>doc/ug_io</i>	html-based documentation files for <i>UG_IO</i> input/output files

Similar directories may be added in the future that follow this same directory structure. The above directories are included in standard package files and contain all files required to run a particular code. Developer packages files are named *_LIB.* and include all the standard package contents along with object code libraries on supported systems, script files, source code for API related routines, and source code for unrestricted code. Restricted or proprietary package files are typically named for a given site, are encrypted, and contain complete source code for a given program or programs. After installing developer or restricted packages the following directories should also exist.

<i>cmake_config/</i>	Developer <i>cmake</i> configuration files, along with numerous CMakeLists.txt files in other applicable directories, such as <i>src/</i> and <i>src/*</i> .
<i>sbin</i>	Developer <i>sh</i> -shell script files.
<i>Linux-x86-64/bin</i>	Executables for applications built for a Linux-x86-64 system.
<i>Linux-x86-64/bin/dlib</i>	Dynamic/shared libraries for applications built for a Linux-x86-64 system.
<i>Linux-x86-64/lib</i>	Libraries of object code for applications built for a Linux-x86-64 system.
<i>arch/bin</i>	Executables for applications built for an <i>arch</i> specific system, such as <i>Linux-x86/bin</i> , <i>MacOSX-x86-64/bin</i> , etc.
<i>arch/bin/dlib</i>	Dynamic/shared libraries for applications built for an <i>arch</i> specific system, such as <i>Linux-x86/bin</i> , <i>MacOSX-x86-64/bin</i> , etc.
<i>arch/lib</i>	Libraries of object code for applications built for an <i>arch</i> specific system, such as <i>Linux-x86/lib</i> , <i>MacOSX-x86-64/lib</i> , etc.
<i>src/ug</i>	Source code for the <i>UG</i> library.
<i>src/...</i>	Source code for other <i>specific</i> libraries, such as <i>ug2</i> , <i>ug3</i> , etc.

4. Configure the SimSys installation.

Script files for developer commands are in the *sbin* developer *sh*-shell script file directory. Developer commands include the following.

<i>simsys_archive</i>	Create package archive of SimSys files.
<i>simsys_compile</i>	Compile a SimSys program or library.
<i>simsys_make_bin</i>	Create a SimSys user bin directory <i>bin/</i> (or optional alternative location) with a copy (or optional link) to all SimSys executables and user <i>sh</i> -shell script files (and optionally developer <i>sh</i> -shell script files).
<i>simsys_www</i>	Create www site package archives for SimSys files (for use at MSU only).

Note that additional user *sh*-shell scripts, such as *simsys_doc*, *simsys_doc.bat*, and *sm*, are also included in the developer script directory for use by *simsys_make_bin* in creation of the user bin directory files. To use the developer commands their path should be included in your system environment PATH. There are two modes available. Choose one of the following operating modes.

- Create a link to all the files in a location that is already in the environment PATH. Change (*cd*) your working directory to the desired directory that you wish to contain the files and run the following command.

```
/path.../simsys/sbin/simsys_make_bin -cwd -link -sbin
```

Where */path...* is the path to the SimSys installation directory. Note that you may also specify a directory rather than moving to it using the option *-dir name* instead of the option *-cwd*. You may need to rerun this command if you create or install new SimSys executables that were not included in the original installation. Links to the executables, once created, will always point to the actual file so that any subsequent executable updates will be readily available. You do not need to rerun the *simsys_make_bin* command each time an executable is updated. A rerun is needed only if a new executable is added that did not exist when you last created the links with *simsys_make_bin* command.

If you wish to create a frozen bin file directory that has copies, rather than links, of the current executable and script files then run the *simsys_make_bin* command in the desired directory and use a *-copy* option instead of the *-link* option. Also, if you don't want the developer commands included then omit the *-sbin* option. These files will be accessible using the version directory option (described later).

- b. Add the locations of the *bin/* and *sbin/* directories to your system environment PATH, e.g. */path.../simsys/bin* and */path to SimSys installation directory.../simsys/sbin*. Where */path...* is the path to the SimSys installation directory.

To run a specific program, use the command line with the name of the specific software followed by options. For example, enter the following command line statement to test the installation and display a usage summary for AFLR3.

```
aflr3 -h
```

If you created a frozen directory or installed other versions of SimSys executables in a location that is not in the system environment PATH then you can access those files using the version directory option. For example, to run another version of AFLR3 that is in directory */other_version.../* then use the following command.

```
aflr3 -v /other_version... [options]
```

Enter the following command line statement to test the developer installation and view options for the compile script.

```
simsys_compile -h
```

For example, to update AFLR3 run this command with *aflr3* as the program name.

```
simsys_compile aflr3
```

This will recompile all AFLR3 related files that have changed and rebuild the executable. The script command *simsys_compile* uses standard make files that are generated by the script to update the program.

5. View documentation.

To view the documentation, it is recommended that you first point your browser to the system documentation main page file */path.../simsys/doc/system/index.html*. The system documentation main page has links to all the overall system related information as well as links to documentation for specific packages. Open a browser and enter this file path or use the command line. A script is also provided to open the system documentation main page. Enter the following command line statement to use this script.

```
simsys_doc
```

Note that if users chose to copy the contents of the *bin/* directory to a directory that is already in their environment PATH (alternative 4. a. above with the *-copy* option) then the script cannot be used as it will not know the actual location of the *doc/* directory (the script assumes that *bin/* and *doc/* have the same root). With a developer installation, this may not be an issue since *simsys_doc* is also in the *sbin/* directory and it will depend on which location is first in the PATH.

6. Additional comments.

All executables created by the *simsys_compile* script are located in the architecture dependent executable directories *arch/bin*. Libraries are in the architecture dependent library directories *arch/bin*. After code is compiled the developer script *simsys_make_bin* can be used to create copies or links to the executables in the user bin directory *bin/*. For a developer installation, it may be advantageous to use links so that there is no duplication.

The executables (and scripts) located in the *bin/* directory use a naming that is the same as that of the program name. On Linux and MacOSX the executables do not have a suffix if a script is not required. On WINDOWS the executables always have an *.exe* suffix. Names and comments for most of the SimSys programs and scripts are listed in the following table.

Linux/MacOSX name	WINDOWS name	Description
<i>simsys_doc</i>	<i>simsys_doc.bat</i>	Script (<i>/bin/sh</i> shell or WINDOWS <i>batch</i> file) to open system main documentation page.
<i>aflr2</i>	<i>aflr2.exe</i>	<i>AFLR2</i> executable (with BL).
<i>aflr2c</i>	<i>aflr2c.exe</i>	<i>AFLR2C</i> executable (with anisotropic adaptation).
<i>bsurf2</i>	<i>bsurf2.exe</i>	<i>BSURF2</i> 2D edge grid creation executable.
<i>xplt2</i>	N/A	<i>XPLT2</i> X-window display for 2D grids executable.
<i>aflr3</i>	<i>aflr3.exe</i>	<i>AFLR3</i> executable.
<i>aflr3_l</i>	<i>aflr3_l.exe</i>	<i>AFLR3</i> executable with 64-bit integers.
<i>aflr4</i>	<i>aflr4.exe</i>	<i>AFLR4</i> executable.
<i>ugc</i>	<i>ugc.exe</i>	<i>UGC</i> executable.
<i>ugc_l</i>	<i>ugc_l.exe</i>	<i>UGC</i> executable with 64-bit integers.
<i>sm</i>	N/A	<i>SolidMesh</i> <i>/bin/sh</i> shell script required to set the <i>SOLID_MESH_SETUP_DIR</i> environment variable. This variable must be set to the location of the interface/ directory, e.g. <i>/path.../simsys/interface/sm</i> . The script can be eliminated if that is made part of your standard environment. The script runs the executable <i>sm.exe</i> . Run <i>SolidMesh</i> using the script name <i>sm</i> or the executable name <i>sm.exe</i> if the <i>SOLID_MESH_SETUP_DIR</i> environment variable is set.
<i>sm.exe</i>	N/A	<i>SolidMesh</i> executable.
<i>checkgrid</i>	N/A	<i>GRID_TOOLS checkgrid</i> executable.
<i>grid_tool_name</i>	N/A	<i>GRID_TOOLS grid_tool_name</i> other executable.
<i>name</i>	<i>name.exe</i>	<i>NAME</i> other SimSys standard executable.