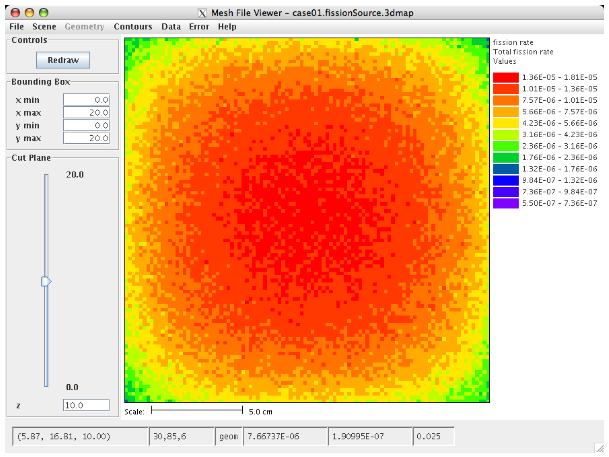
Getting Started with SCALE 6.1

Introduction

The Standardized Computer Analysis for Licensing Evaluation (SCALE) code system developed at Oak Ridge National Laboratory provides a comprehensive, verified and validated, user-friendly tool set for criticality safety, reactor physics, spent fuel characterization, radiation shielding, and sensitivity and uncertainty analysis. Since 1976, regulators, licensees, and research institutions around the world have used SCALE for safety analysis and design. SCALE provides a “plug-and-play” framework with nearly 80 computational modules, including three deterministic and three Monte Carlo radiation transport solvers that are selected based on the desired solution. SCALE’s graphical user interfaces assist with accurate system modeling and convenient access to desired results. SCALE 6.1 provides improved reliability and introduces a number of enhanced features in a robust yet user-friendly package that are intended to improve safety and efficiency throughout the nuclear community.

CRITICALITY SAFETY

The KENO Monte Carlo neutron transport codes for eigenvalue problems have realized a number of enhancements for SCALE 6.1. The mesh flux and fission source accumulator used in TSUNAMI-3D sensitivity analysis sequences and in criticality accident alarm system analysis has been improved with more flexibility in the user definition of mesh intervals and better mesh volume calculations, mesh tracking, and output edits. Mesh fission source data can now be generated using KENO V.a or KENO-VI in multigroup or continuous-energy mode, and the fission distribution can be visualized with the MeshView tool, as shown in Fig. 1. Default criticality search parameters have been modified to provide improved convergence to true minimum or maximum values, and region mean free paths can now be computed in continuous-energy mode.



x min 0.0
x max 20.0
y min 0.0
y max 20.0
z 10.0

fission rate
Total fission rate
Values

🟥	1.36E-05 - 1.81E-05
🟥	1.01E-05 - 1.36E-05
🟧	7.57E-06 - 1.01E-05
🟧	5.66E-06 - 7.57E-06
🟨	4.23E-06 - 5.66E-06
🟩	3.16E-06 - 4.23E-06
🟩	2.36E-06 - 3.16E-06
🟩	1.76E-06 - 2.36E-06
🟦	1.32E-06 - 1.76E-06
🟦	9.84E-07 - 1.32E-06
🟪	7.36E-07 - 9.84E-07
🟪	5.50E-07 - 7.36E-07

```
(5.87, 16.81, 10.00)   30,85,6   geom   7.66737E-06   1.90995E-07   0.025
```

Fig. 1. Mesh fission source from KENO V.a visualized in MeshView.

SHIELDING ANALYSIS

The MAVRIC shielding analysis capabilities with automated variance reduction capabilities, first introduced with SCALE 6.0, have realized a number of enhancements for their second release in SCALE 6.1. Multiple sources may now be defined with spatial distributions defined within each source. Energy distributions can be imported from an ORIGEN binary concentration file or from response functions read from an AMPX cross-section file. MAVRIC also includes improvements in the advanced variance reduction capabilities such as a macro materials option for improved Denovo deterministic simulations used to generate variance reduction parameters and increased flexibility in forward-weighting strategies. Cylindrical mesh grids have been added to more accurately capture spatial effects, as shown in Fig. 2, for a shielding calculation for a spent fuel shipping cask, and a suite of MAVRIC utilities has been developed to postprocess data files.

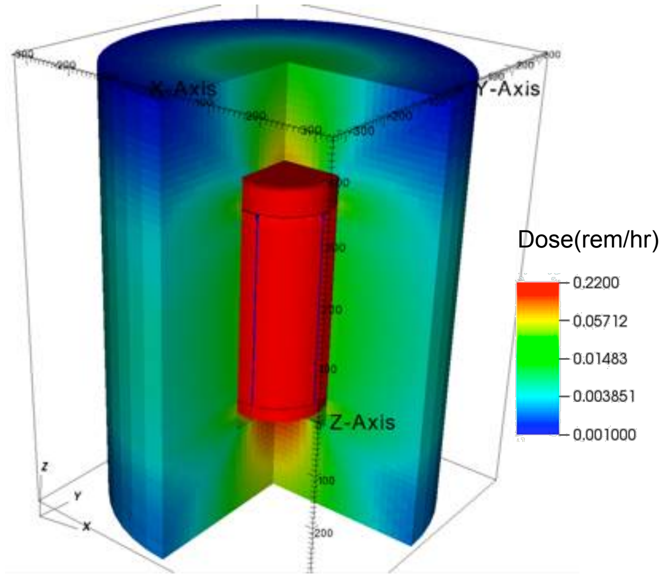


Fig. 2. MAVRIC spent fuel shipping cask model with a cylindrical mesh tally showing dose in rem/hr.

DEPLETION AND DECAY

The ORIGEN and COUPLE codes used for isotopic activation, depletion and decay have been substantially improved for SCALE 6.1. Support is now provided for multigroup cross-section libraries in any group structure, ENDF/B-VII decay libraries, and energy-dependent fission product yields. Cross-section transitions can be included from multiple sources, including JEFF-3.0/A-based AMPX format multigroup cross-section libraries developed for burnup and activation applications, an AMPX library generated by one of the SCALE transport codes, and cross sections input manually by the user via the input file.

REACTOR PHYSICS

The TRITON reactor physics capabilities have realized a number of enhancements for SCALE 6.1. The KENO-based Monte Carlo depletion capabilities have been substantially improved to more accurately compute power distributions and enable all KENO functionalities such as source specification, region volume input, and geometry plotting. TRITON was updated to use the improved multigroup functionality of ORIGEN and COUPLE, and for clusters with multiple computing nodes, branch calculations can now be run in parallel.

The two-dimensional (2D) generalized geometry lattice physics code NEWT was enhanced with parallel operation, support for inhomogeneous sources for generalized perturbation theory (GPT) calculations, improved support for high temperature gas reactor prismatic geometries, and support for coupled n-gamma calculations. Several corrections were also realized, including improved unstructured course-mesh finite diffusion acceleration, grid generation algorithms, results for few-group homogenized cross sections, and output edits.

SENSITIVITY AND UNCERTAINTY ANALYSIS

For SCALE 6.1 the TSUNAMI-3D adjoint-based sensitivity and uncertainty analysis capabilities were enhanced through many of the previously described improvements in the KENO mesh capabilities, and a new TSUNAMI-2D capability was introduced using NEWT as the transport solver. In addition to generating the sensitivity of k_{eff} and reactivity to the multigroup cross-section data, one-dimensional and 2D capabilities were introduced for GPT calculations, expanding the responses for which sensitivities and uncertainties can be computed to include flux ratios, reaction rate ratios, and collapsed few-group cross sections. The TSURFER data assimilation code for advanced bias and bias uncertainty assessment was updated with improved output edits and plots.

NUCLEAR DATA

The SCALE nuclear data files have also been enhanced. For uranium and plutonium isotopes in ENDF/B-VI.8 and ENDF/B-VII.0 continuous-energy cross sections, the unresolved resonance region probability tables have been improved, providing more accurate results, especially for intermediate energy systems.

The 238-group ENDF/B-VI.8 and ENDF/B-VII.0 neutron criticality libraries have been updated with an improved weighting function in which the tie-in for the fission spectrum has been raised from 67.4 to 820.8 keV. This adjustment improved the performance of spectral calculations for very high temperature reactor simulations. In addition, updated versions of AMPX routines using double precision throughout the calculation were used for the library generation.

The ORIGEN data have been updated to include ENDF/B-VII decay and fission yield libraries and JEFF multigroup neutron cross-section libraries in 44-, 47-, 49-, 200-, and 238-group structures. The new decay library includes 2227 nuclides, including 174 actinides, 1149 fission products, and 904 structural activation materials.

GRAPHICAL USER INTERFACES

Many of the SCALE graphical user interfaces have been enhanced for SCALE 6.1. Notably, the GeeWiz input interface for Windows now fully supports all major SCALE computational sequences and provides a more stable and intuitive work environment. The Javapeño data visualization package now supports plotting SCALE continuous-energy data and ORIGEN data output from the OPUS module.

System Requirements

System Architecture:

- Linux 32bit and 64bit
- Darwin 9 and 10
- Windows XP, Vista and 7

Required Memory:

Minimum of 2 GB, Recommended 4 GB of RAM

Minimum 30 GB available disk space

Java Requirements:

Java1.6

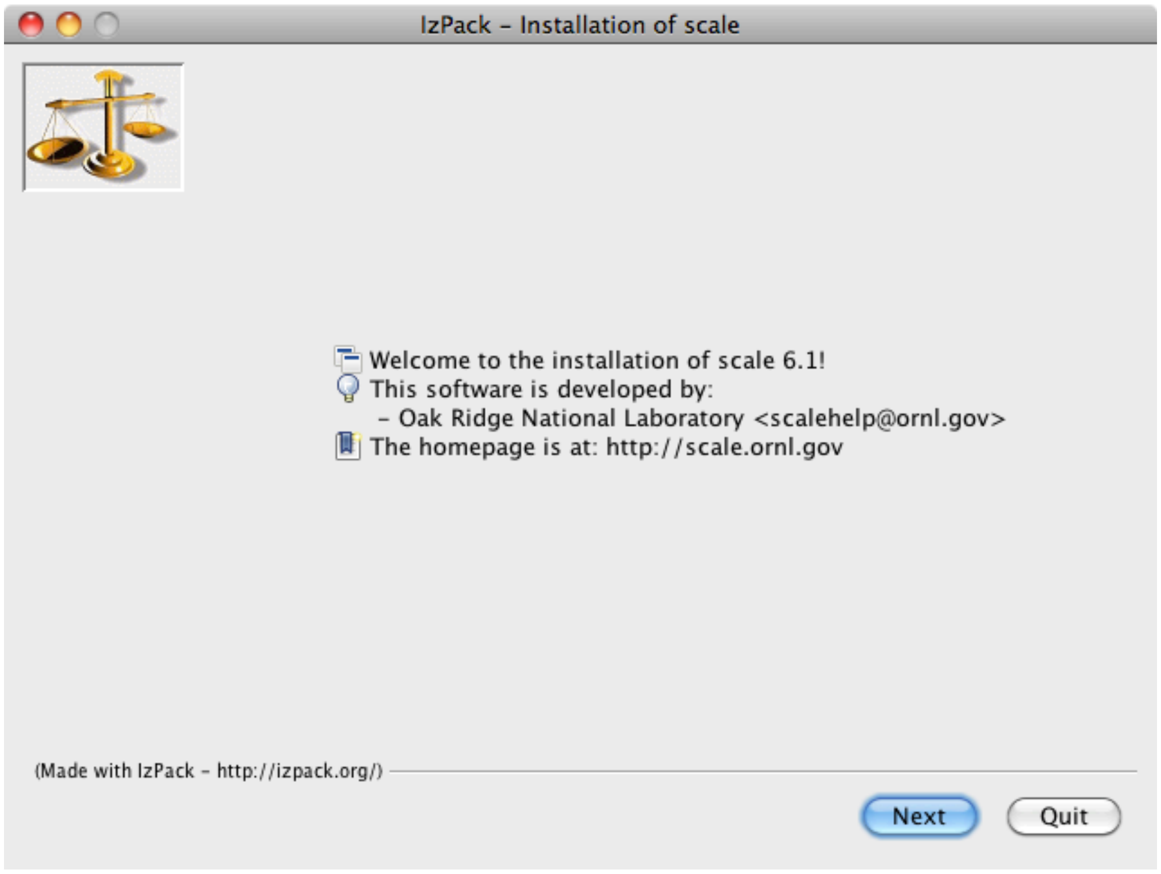
Java3D 1.5

Installation Instructions

SCALE 6.1 includes an installer that provides an interactive interface for full or customizable installations.

To begin installation of SCALE 6.1 if you are on Windows, double-click the *scale-6.1-setup.jar* file, if you are on Linux or Mac copy the scale-6.1-setup.jar to your local disk and double-click the local version.

This will open an installer dialog.



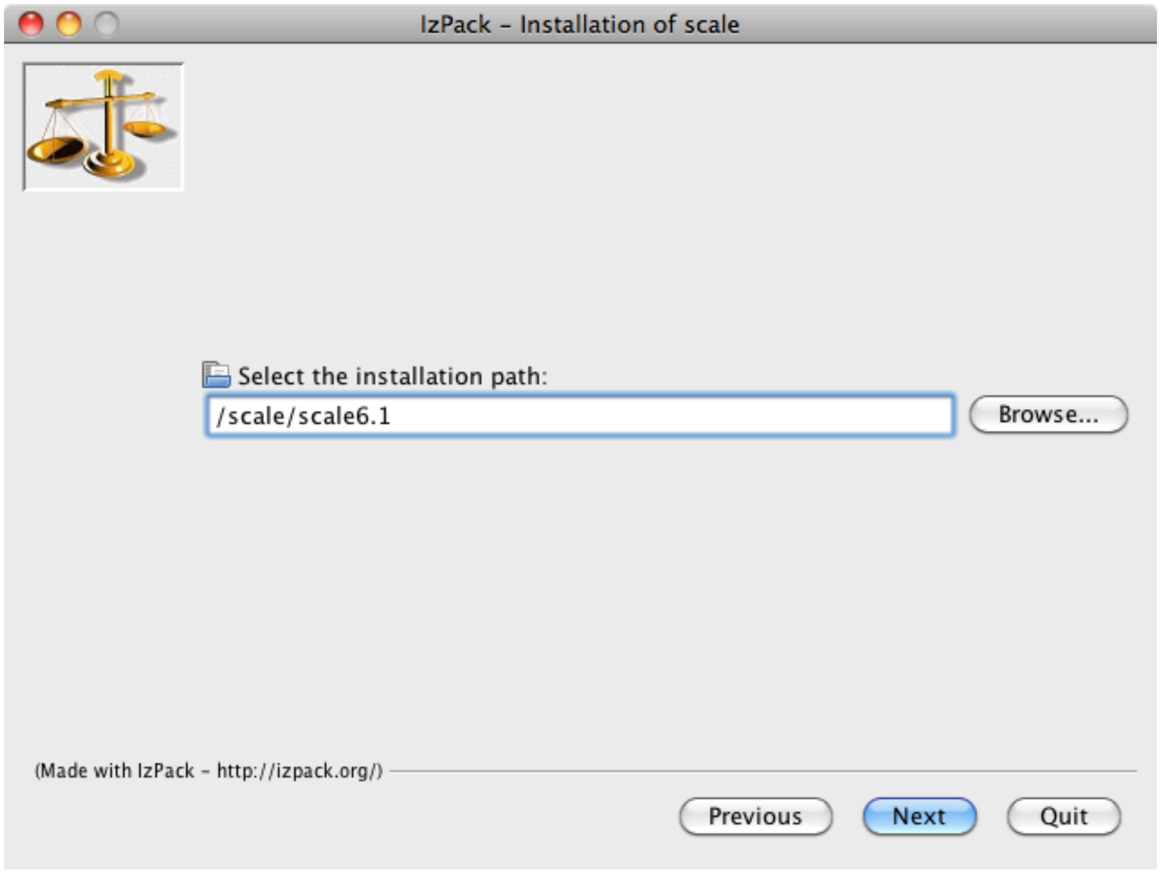
IzPack – Installation of scale

Welcome to the installation of scale 6.1!
This software is developed by:
– Oak Ridge National Laboratory <scalehelp@ornl.gov>
The homepage is at: http://scale.ornl.gov

(Made with IzPack – http://izpack.org/)

Please follow the steps by pressing 'Next'.

You will be prompted to choose the destination of your installation.



Select the installation path:

/scale/scale6.1

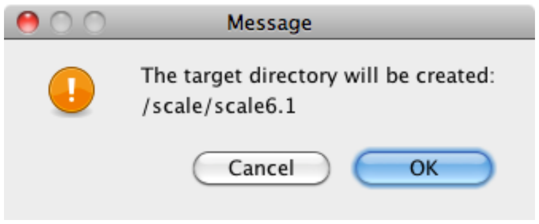
(Made with IzPack - http://izpack.org/)

For Windows users, the recommended installation path is at any root level(c:\scale6.1 ,d:\scale6.1, etc...)

This is highly recommended as the Windows graphical user interfaces GeeWiz, OrigenArp, Keno3d and PlotOpus require scale to be installed at the root level.

For Linux and Mac systems, a typical location is /scale/scale6.1.

The installer will present a prompt to confirm the creation of a new directory, press OK.

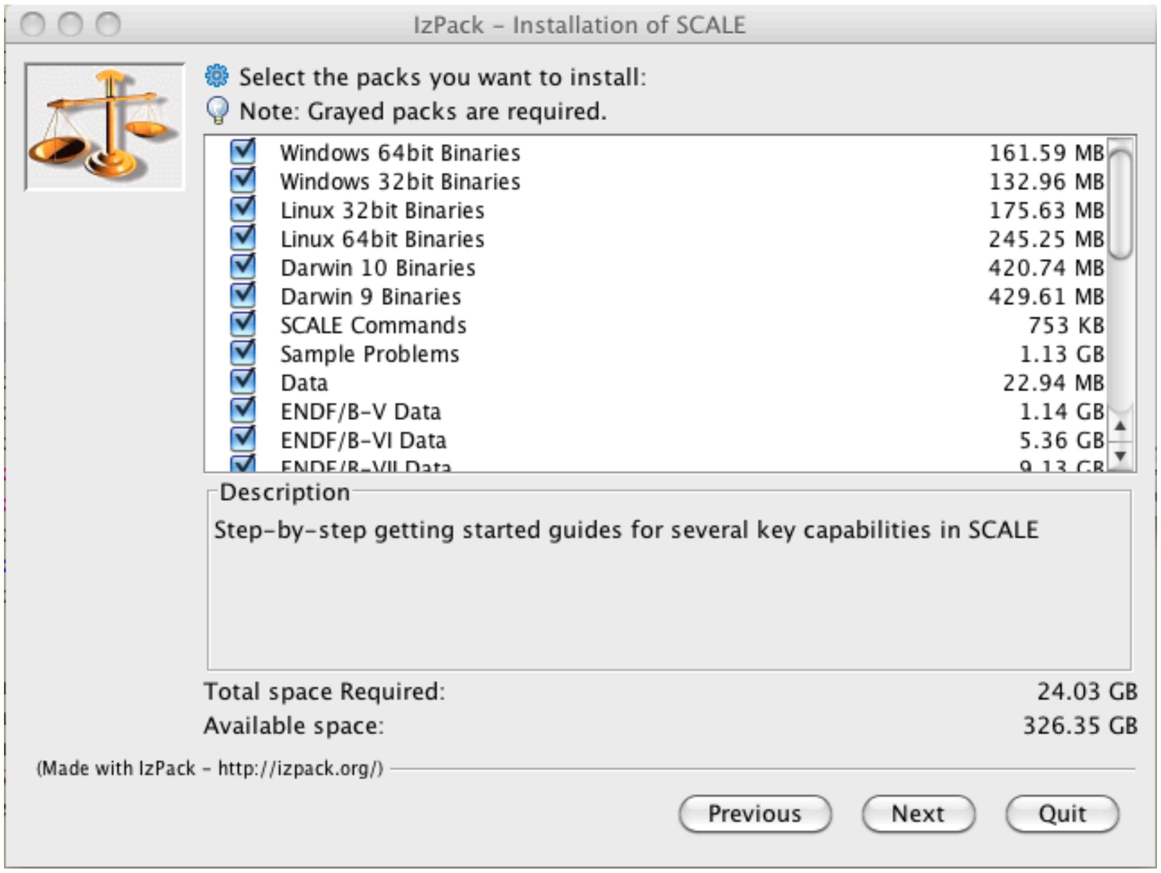


The target directory will be created:
/scale/scale6.1

*If the installer prompts you that "C:\scale6.1" directory can not be created, click browse, create the new folder "C:\scale6.1" and select it.

Once you select 'Next' it will prompt you that the directory already exists. Click 'Ok' to overwrite the directory and click 'Next'.

The installer then presents all available options for installation. By default, SCALE installs all available options. The only two required options are SCALE Binaries, the executable files for SCALE, and SCALE Data, the data files for SCALE.


IzPack - Installation of SCALE

Select the packs you want to install:

Note: Grayed packs are required.

✓	Windows 64bit Binaries	161.59 MB
✓	Windows 32bit Binaries	132.96 MB
✓	Linux 32bit Binaries	175.63 MB
✓	Linux 64bit Binaries	245.25 MB
✓	Darwin 10 Binaries	420.74 MB
✓	Darwin 9 Binaries	429.61 MB
✓	SCALE Commands	753 KB
✓	Sample Problems	1.13 GB
✓	Data	22.94 MB
✓	ENDF/B-V Data	1.14 GB
✓	ENDF/B-VI Data	5.36 GB
✓	ENDF/B-VII Data	9.13 GB

Description

Step-by-step getting started guides for several key capabilities in SCALE

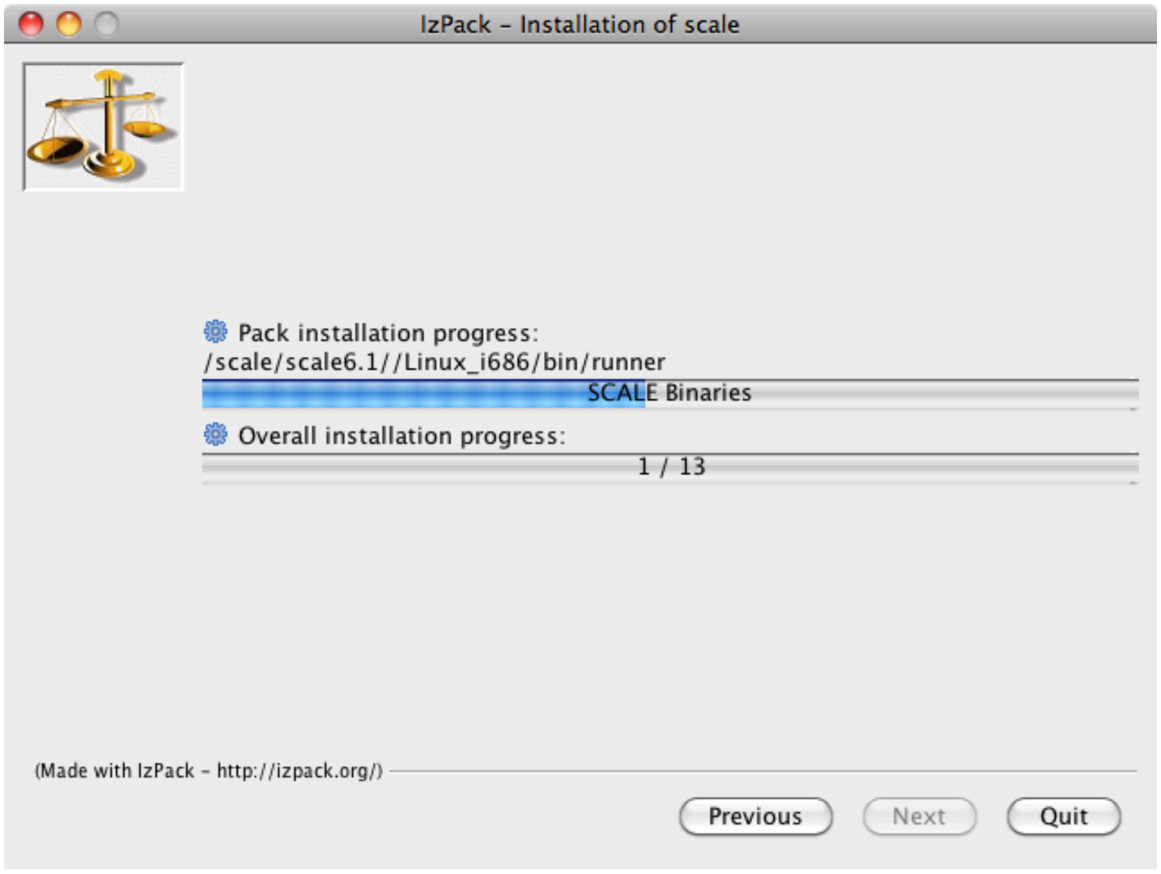
Total space Required: 24.03 GB

Available space: 326.35 GB

(Made with IzPack - http://izpack.org/)

After selecting the desired options, press *next* to proceed.

If you are running on Linux or Mac you will be prompted to select ‘installer.pak’. Please navigate to your DVD and select ‘installer.pak’ and proceed.


IzPack – Installation of scale
Pack installation progress:
/scale/scale6.1//Linux_i686/bin/runner
SCALE Binaries
Overall installation progress:
1 / 13
(Made with IzPack – http://izpack.org/)
Previous
Next
Quit

The installer will post the progress of the installation. Part way through the installation you will be prompted for 'installer.pak.1' please insert the PAK.1 disk and click 'Apply'.

PAK.1 will complete and prompt for 'installer.pak.2' please insert the PAK.2 disk and click 'Apply'.

PAK.2 will finish and you have completed your installation of SCALE 6.1.

If you have any questions please contact scalehelp@ornl.gov

Build Instructions

These instructions are only for those who wish to recompile their SCALE binaries. If you do not need to recompile your SCALE binaries please skip this section.

SCALE required the following programs in order to compile:

1. Intel Fortran 11.1+ compiler.
2. GNU g++/gcc 4.2+ compiler
3. Openmpi mpif90 compiled with Intel Fortran. (Only for Nix systems)
4. CMake - Platform independent build configuration.

SCALE requires the following libraries:

1. Trilinos libraries:
    a. Amesos
    b. Anasazi
    c. Aztecoo
    d. Epetra
    e. Epetraext
    f. Teuchos
    g. Triutils
2. LAPACK libraries:
    a. lapack
    b. blas

SCALE RUNNER requires:

mpif90 compiled with ifort. You will need to download the openmpi source code, recompile and configure the build process with the following:

```
./configure F90=ifort F77=ifort FC=ifort
```

If you are using 64bit, add 64bit flags to the other compilers.

Example:

```
./configure F90=ifort F77=ifort FC=ifort  CC="gcc -m64" CXX="g++ -m64"
```

OVERVIEW

There are three main steps before you have SCALE binaries

1. CMake configuration - This generates a native build tree
2. Compilation - This compiles all executables and libraries
3. Installation - This deploys all executables into a configuration ready for running

CMake Configuration

Throughout SCALE you will find 'CMakeLists.txt' files. From the SCALE root directory these CMakeLists.txt files create a tree of included directories called the SOURCE TREE. Namely the source directories:

src/aim

    bonami

    scalelib

    etc...

To configure you build you call 'cmake' on the root CMakeLists.txt file, namely scale_dir/CMakeLists.txt. CMake takes your source tree and creates a BUILD TREE. The build tree contains or will contain your build configuration, Make or NMake files, and all compilation output, i.e., object files, archive libraries and binary executables.

SCALE requires several third party libraries (TPL).

Namely: Trilinos and Lapack. These TPL libraries and includes must be specified at the time of configuration. For ease of use configuration scripts for every supported platform are available in the 'scripts' directory of your checkout. These scripts describe the necessary variables that need to be defined. A user’s modifications to these scripts should be limited to the path to the root directories for the TPLs.

Recommended Configuration Procedure

After checking out the source code, navigate to the root scale directory. You should see CMakeTPL.txt, CMakeLists.txt and CMakeConfig.txt. This is the root of the source tree, what you will point CMake at. This example will demonstrate creating build trees for multiple architectures for your working copy.

Make build directory

mkdir build

mkdir build/Linux_x86_64 **this could be any directory, I want to emphasize multiple platform builds.

or

mkdir build/Windows_amd64 **for purposes of this example I will use Linux_x86_64

CMake Initialization

Move the cmake script from the 'scripts' directory to your build directory

```
cp scripts/linux_x86_64-cmake build/Linux_x86_64

chmod u+x build/Linux_x86_64/linux_x86_64-cmake
```

/** At this point your may need to update the TPL paths in this script **/

Create Your Configuration

Create your configuration by pointing this script at the source tree root

```
cd build/Linux_x86_64

./linux_x86_64-cmake ../..

/*** Configuration Output.... ***/
```

**Things to know are that if you add or remove source files from the source tree, cmake will NOT see these modifications unless it modified a CMake file(CMakeLists.txt,CMakeTPL.txt,CMakeConfig.txt...). CMake will re-evaluate the entire source tree when any CMake file has been modified. If CMake does not pick up the add/removal of sources files the easiest way to update your build tree is to 'touch' any CMake file in the source tree.

Compilation

Every library and executable is a TARGET. Calling 'make', 'nmake' on windows, from the root of your build tree, build/Linux_x86_64 from the previous example, will build ALL targets. For building specific targets there are two methods.

For example mavricUtilities contains a number of executables: mtadder, mtaverager, etc...

Either you may 'make mtadder' to build only mtadder or you can, from build/Linux_x86_64, 'cd src/mavricUtilities' and 'make' which will compile ALL targets in mavricUtilities.

Compilation Flags

CMakeConfig.txt contains the logic for detecting the current platform and setting the appropriate compilation flags for the ENTIRE build tree. You may modify CMAKE_Fortran_FLAGS or CMAKE_C_FLAGS in CMakeConfig.txt, which results in a global effect or you may set these variables on a per module basis.

Installation

CMake provides the 'install' target, which installs all binaries from the current directory down. For example 'make install' from build/Linux_x86_64 will install any targets in src down, which would install aim, mavric, mtadder, mtaverager etc... You can also 'cd src/aim' and make install will only install declared in the aim directory.

***Things to know are that 'make install' re-evaluated the build for all dependencies. Take for example package A which depends on packages B and D. Package D depends on package E. 'make install' for package A would result in packages E, D and B being re-evaluated and rebuilt if necessary. Thus if you know you want to build and install, you can save time building by simply doing a 'make install'.

Runtime Environment

The SCALE runtime environment *scalerte* has been re-written to be platform independent and more self-aware. scalerte provides for easier setup and faster methods to get started with SCALE 6.1, while keeping backwards compatibility with previous practice. Written in C++ and XML to provide performance oriented, versatile, and portable usage, it allows SCALE to be run in configurations previously not available. This section describes the environment provided by scalerte, the arguments available, and typical usage.

Running SCALE

* Note that Norton Antivirus may require an exception for scalerte.exe on Windows

The SCALE runtime environment provides several command line options. The platform specific entry point into SCALE is the batch6.1 scripts: *batch6.1* bash script on Linux and Mac (*Nix* systems), and *batch6.1.bat* on Windows. The usage is as follows.

```
batch6.1 [options] inputfile(s) [options] [inputfile(s)]
```

Where `inputfile(s)` are one or more files or file patterns (*test.inp*, or *test*.inp*, etc.).

Where *options* are:

`-a`: Specify alias file.

```
-a path/to/aliasesfile
```

`-d`: Specify the SCALE driver directory, relative to SCALE

`($SCALE\\YourDirectory)`.

```
-d dir/contains/binaries
```

`-f`: Add hostname to output filename. Produces inputfile.hostname.out

`-h`: Print this information as a help message

`-I`: Number of threads to use for MPI/OpenMP directives. `-I 4"`

`-m`: Print information messages as SCALE executes

`-M`: Specify a machine names file for SCALE parallel capabilities.

```
-M /path/to/machine/names/file
```

`-n`: Nice level on Nix systems, ignored on Windows. Default: `-n 2`

`-N`: Number of nodes on which to execute MPI directives. `-N 20`

`-o`: Overrides the default *inputfile.out* output name. The *.out* extension is appended by scalerte, so there is no need to specify the extension.

```
-o path/to/outputfile
```

NOTE: If the *path/to/outputfile* already exists, it will be deleted. If this option is specified while in stack mode (multiple input files), the value provided is prepended to the inputfile's basename.

`batch6.1 triton* -o myout` results in output names `myout.triton*.out`

`-p`: Set block letters. This prints ASCII art banners into your output file.

`-r`: Keep the working directory after execution

`-s`: Overrides platform directory. `-s Linux_x86_64`

`-t`: No new temporary directory. Uses last temporary directory, `-r` is implied.

`-T`: Specify working directory.

```
-T directory/path
```

NOTE: If `-T` is specified while in stack mode (multiple input files) the value provided is appended with the index of the file.

```
batch6.1 t0.inp t1.inp t2.inp -T mytmp
```

results in *`mytmp`*, *`mytmp1`*, and *`mytmp2`* temporary directories.

`-v`: Turn on verbose activity printing for scalerte.

`-V`: Print the scalerte version date.

`-x`: Do not return XSDRNPM output in a *.xsdrnfiles* directory.

`-z`: Add date to the output filename.

Produces output files in the form of
`inputfile.yyyy.MM.ddThh.mm.ss.out`

Where:

- `yyyy- is the year of execution.`
- `MM- is the month of execution.`
- `dd- is the day of execution.`
- `hh- is the hour of execution.`
- `mm- is the minute of execution.`

- `ss- is the second of execution.`

Example Invocation

For users familiar with previous invocation of the SCALE batch script, usage can remain the same. Note that options can be specified both before and after the input file(s). Note that the examples below assume that batch6.1 is called from the SCALE 6.1 *cmds* directory or that the SCALE 6.1 *cmds* directory is in your system path.

Invoke SCALE on a single input file named *HelloWorld.inp*

```
batch6.1 HelloWorld
```

or

```
batch6.1 HelloWorld.inp
```

Invoke SCALE on all input files patterned *HelloWorld*.inp*

```
batch6.1 HelloWorld*.inp
```

Invoke SCALE on all input files patterned *HelloWorld*.inp* and print runtime messages to the console

```
batch6.1 HelloWorld*.inp -m
```

Invoke SCALE on all input files patterned *HelloWorld*.inp* and include hostname and date/time in the output file's name

```
batch6.1 HelloWorld*.inp -fz
```

or

```
batch6.1 HelloWorld*.inp -f -z
```

Invoke SCALE on *HelloWorld.inp* and rename output to be *MyHello.out*

```
batch6.1 HelloWorld -o MyHello
```

or

```
batch6.1 HelloWorld.inp -o MyHello
```

Invoke SCALE on all files patterned *HelloWorld*.inp* and rename output to be *MyHelloWorld*.out*.

NOTE: When SCALE is run in stack mode (multiple inputs), the output override is prepended to the input file's name.

```
batch6.1 HelloWorld*.inp -o My
```

Invoke SCALE on *HelloWorld.inp* and keep the working directory.

```
batch6.1 HelloWorld.inp -r
```

Invoke SCALE on *HelloWorld.inp* and override and keep the working directory.

```
batch6.1 HelloWorld.inp -r -T myHelloWorldTempDir
```

Invoke SCALE on *HelloWorld.inp* and specify the number of threads to be 4.

```
Batch6.1 HelloWorld.inp -I 4
```

SCALE Sample Problems

The SCALE sample problems are carefully designed by the SCALE developers to verify the installation and functionality of SCALE relative to expected results. scalerte has a built in scripting interface to allow for invocation of several groupings of sample problems, from individual problems, problems for a specific module, 'short' subset, or all samples. The sample problems will print a message indicating the currently running sample problem, followed by the differences between ORNL-generated results and the newly generated results for the sample problem(s).

To invoke all sample problems do the following.

```
batch6.1 @samples
```

Every sample problem is exposed as a target, as well as all are grouped for convenience.

To invoke the 'short' subset of the sample problems, do the following.

```
batch6.1 @samples short
```

To invoke sample problems for a particular module or sequence, use

```
batch6.1 @samples modulename
```

where `modulename` is the name of the module or sequence to test, e.g. *centrm*

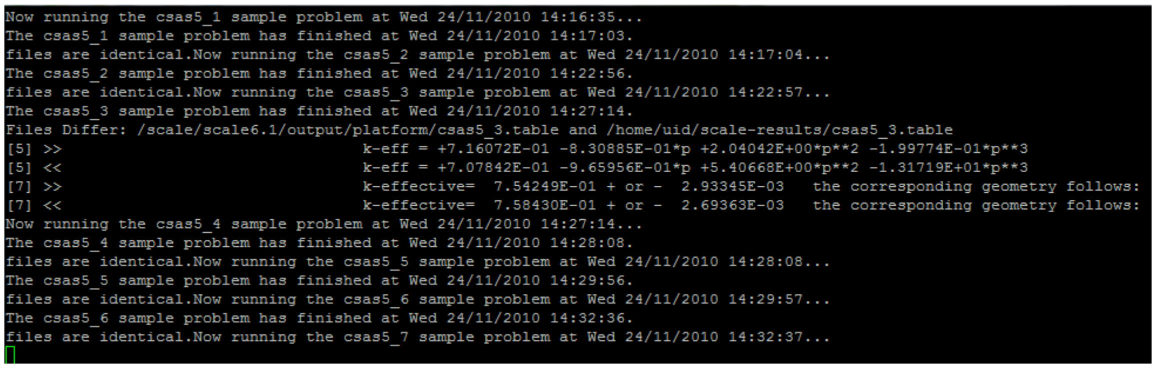
To invoke a single sample problem sample problem, do the following.

```
batch6.1 @samples problemname
```

where `problemname` is the name of the specific sample problem to test, e.g. *centrm-pwr*

Because the target system in the scripting layer is based on words contained in a target's category attribute, users can, for example, execute all CSAS sample problems (cecsas5, cecsas6, csas5, csas6) by doing the following.

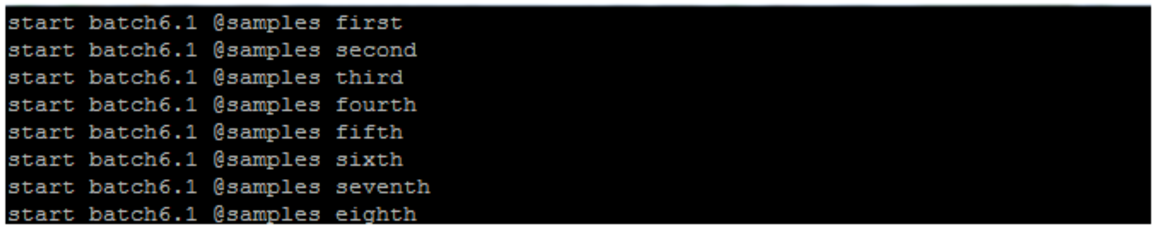
```
batch6.1 @samples csas
```



```
Now running the csas5_1 sample problem at Wed 24/11/2010 14:16:35...
The csas5_1 sample problem has finished at Wed 24/11/2010 14:17:03.
files are identical.Now running the csas5_2 sample problem at Wed 24/11/2010 14:17:04...
The csas5_2 sample problem has finished at Wed 24/11/2010 14:22:56.
files are identical.Now running the csas5_3 sample problem at Wed 24/11/2010 14:22:57...
The csas5_3 sample problem has finished at Wed 24/11/2010 14:27:14.
Files Differ: /scale/scale6.1/output/platform/csas5_3.table and /home/uid/scale-results/csas5_3.table
[5] >>                          k-eff = +7.16072E-01 -8.30885E-01*p +2.04042E+00*p**2 -1.99774E-01*p**3
[5] <<                          k-eff = +7.07842E-01 -9.65956E-01*p +5.40668E+00*p**2 -1.31719E+01*p**3
[7] >>                          k-effective=  7.54249E-01 + or -  2.93345E-03   the corresponding geometry follows:
[7] <<                          k-effective=  7.58430E-01 + or -  2.69363E-03   the corresponding geometry follows:
Now running the csas5_4 sample problem at Wed 24/11/2010 14:27:14...
The csas5_4 sample problem has finished at Wed 24/11/2010 14:28:08.
files are identical.Now running the csas5_5 sample problem at Wed 24/11/2010 14:28:08...
The csas5_5 sample problem has finished at Wed 24/11/2010 14:29:56.
files are identical.Now running the csas5_6 sample problem at Wed 24/11/2010 14:29:57...
The csas5_6 sample problem has finished at Wed 24/11/2010 14:32:36.
files are identical.Now running the csas5_7 sample problem at Wed 24/11/2010 14:32:37...
```

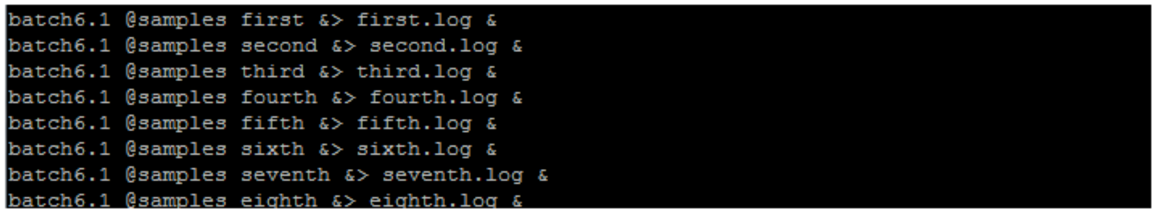
Built into the target system is a concept of groups, maximum of 8. This provides for a convenient way by which to run groups of sample problems concurrently. If you have 8 processors you can run each group concurrently with the following:

```
Windows console:
```



```
start batch6.1 @samples first
start batch6.1 @samples second
start batch6.1 @samples third
start batch6.1 @samples fourth
start batch6.1 @samples fifth
start batch6.1 @samples sixth
start batch6.1 @samples seventh
start batch6.1 @samples eighth
```

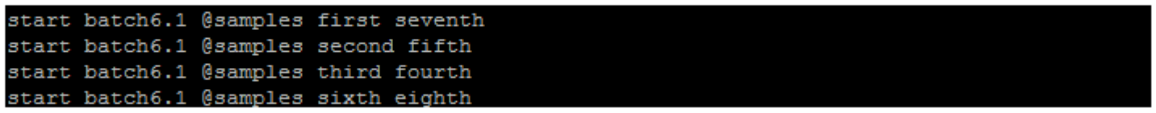
```
Linux and Mac console:
```



```
batch6.1 @samples first &> first.log &
batch6.1 @samples second &> second.log &
batch6.1 @samples third &> third.log &
batch6.1 @samples fourth &> fourth.log &
batch6.1 @samples fifth &> fifth.log &
batch6.1 @samples sixth &> sixth.log &
batch6.1 @samples seventh &> seventh.log &
batch6.1 @samples eighth &> eighth.log &
```

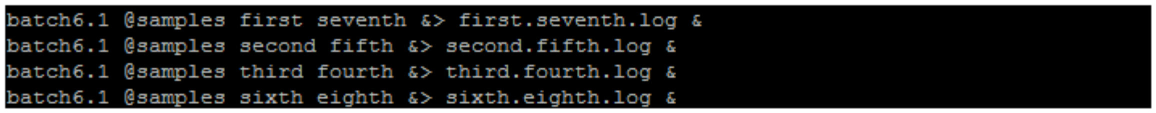
If your computer does not have 8 processors, we can stack the groups to accommodate arbitrary number of processors. Here is an example of running the groups on 4 processors:

```
Windows console:
```



```
start batch6.1 @samples first seventh
start batch6.1 @samples second fifth
start batch6.1 @samples third fourth
start batch6.1 @samples sixth eighth
```

```
Linux and Mac console:
```



```
batch6.1 @samples first seventh &> first.seventh.log &
batch6.1 @samples second fifth &> second.fifth.log &
batch6.1 @samples third fourth &> third.fourth.log &
batch6.1 @samples sixth eighth &> sixth.eighth.log &
```

For any problem or questions, please contact scalehelp@ornl.gov.

SCALE Variables

This section describes the environment variable used within scalerte. These variables can be access through SCALE's *shell* module to populate the working directory, and/or to return SCALE generated files that are not returned by scalerte.

There are seven primary locations known by scalerte. These primary locations are:

-The user's home directory, HOME.

*Nix systems, $HOME, /home/uid.

*Windows, %HOME%, C:\Users\uid.

- The directory of SCALE, SCALE.

*Nix systems, $SCALE, location of user's installation. Typically /scale/scale#.

*Windows, %SCALE%, location of user's installation. Typically C:\Scale#.

- The directory of the input file, INPDIR.

*Nix systems, $INPDIR.

*Windows, %INPDIR%.

- The directory of the output file, OUTDIR, by default, this is the same as INPDIR, because the output file is written next to the input file.

*Nix systems, $OUTDIR.

*Windows, %OUTDIR%.

- The directory from which SCALE was invoked, the return directory, RTNDIR. This is the directory your console will return to upon completion.

*Nix systems, $RTNDIR.

*Windows, %RTNDIR%.

- The directory that contains the SCALE data, DATA.

*Nix systems, $DATA.

*Windows, %DATA%.

- The working directory for a given input file, TMPDIR, or shorthand TMP.

*Nix systems, $TMPDIR, $TMP.

*Windows, %TMPDIR%, %TMP%.

There are several secondary locations, located in the SCALE directory tree. These are as follows:

-The directory containing the command scripts associated with SCALE, CMDS.

*Nix systems, $CMDS.

*Windows, %CMDS%.

-The directory containing the platform-specific compiled programs, PGMDIR, or legacy PGM_DIR.

*Nix systems, $PGMDIR, $PGM_DIR.

*Windows, %PGMDIR%, %PGM_DIR%.

Lastly, there are several environment variables provided for convenience, and/or associated with output data that can be useful.

-The directory containing the ORIGEN data files, ORIGENDIR.

*Nix systems, $ORIGENDIR.

*Windows, %ORIGENDIR%.

-The base name of the input file, BASENAME. This is the name of the input file without both absolute path and extension.

*Nix systems, $BASENAME, or $CASE_NAME.

*Windows, %BASENAME%, or %CASE_NAME%.

-The base name of the output file, OUTBASENAME, or legacy CASE_NAME. This is the name of the output file without both absolute path and extension.

*Nix systems, $OUTBASENAME, or $CASE_NAME.

*Windows, %OUTBASENAME%, or %CASE_NAME%.

-The base name of the output file, OUTBASE. This is the absolute name of the output file without absolute file extension.

*Nix systems, $OUTBASE.

*Windows, %OUTBASE%.

-The absolute path to the input file, INPUTFILE.

*Nix systems, $INPUTFILE.

*Windows, %INPUTFILE%.

-The absolute path to the output file, OUTFILE.

*Nix systems, $OUTFILE.

*Windows, %OUTFILE%.

- The directory containing SMORES output, SMORESDIR. If SMORES data were output, these data will be located in OUTDIR\OUTBASENAME.outfiles directory.

*Nix systems, $SMORESDIR.

*Windows, %SMORESDIR%.

- The directory containing USLSTATS output, USLSTATSDIR. If USLSTATS data were output, these data will be located in OUTDIR\OUTBASENAME.uslstats directory.

*Nix systems, $USLDIR.

*Windows, %USLDIR%.

- The directory containing CENTRM output, CENTRMDIR. If CENTRM data were output, these data will be located in OUTDIR\OUTBASENAME.centrmfiles directory.

*Nix systems, $CENTRMDIR.

*Windows, %CENTRMDIR%.

- The directory containing XSDRNPM output, XSDRNDIR. If XSDRNPM data were output, these data will be located in OUTDIR\OUTBASENAME.xsdrnfiles directory.

*Nix systems, $XSDRNDIR.

*Windows, %XSDRNDIR%.

-The platform specific file separator, FS. This is either back slash (\) on Windows, or forward slash (/) on Nix systems.

*Nix systems, $FS.

*Windows, %FS%.