# Swiftrun

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# 1 Introduction

Swift 0.95 introduces many changes to the method for configuring and running Swift. The goal of these changes is to make configuration easier for Swift users. This document will attempt to explain the reasons for these changes and document how these new configuration mechanisms work.

# 2 Running older Swift releases

In Swift 0.94 and earlier versions, configuring Swift usually required setting up files called sites.xml, tc.data, and cf. These files typically needed to be specified on the command line. The following command line was pretty typical in previous Swift releases:

```
$ swift -sites.file sites.xml -tc.file tc.data -config cf myscript.swift
```

## 2.1 sites.xml

The sites.xml file was an XML configuration file that defined site parameters. It was used to determine how Swift should interact with the scheduler. Below is an example sites.xml for a campus cluster called midway

**sites.xml**

```
<config>
  <pool handle="midway-sandyb">
    <execution provider="coaster" jobmanager="local:slurm"/>
    <profile namespace="globus" key="jobsPerNode">16</profile>
    <profile namespace="globus" key="maxWalltime">_WALLTIME_</profile>
    <profile namespace="globus" key="highOverAllocation">100</profile>
    <profile namespace="globus" key="lowOverAllocation">100</profile>
    <profile namespace="globus" key="queue">sandyb</profile>
    <profile namespace="karajan" key="initialScore">10000</profile>
    <filesystem provider="local"/>
    <workdirectory>/scratch/midway/{env.USER}</workdirectory>
  </pool>
</config>
```

## 2.2 tc.data

The tc.data was a basic catalog that defined the location of applications on a given site.

**tc.data**

```
localhost  cat  /bin/cat null null null
```

## 2.3 cf

The cf file (also called swift.properties) was a file that defined various swift configuration values, like retries and error handling.

**cf**

```
wrapperlog.always.transfer=true
sitedir.keep=true
file.gc.enabled=false
status.mode=provider
```

# 3 Running Swift 0.95

Previous versions of Swift required users to create multiple files, each in stored in a different format. In an attempt to make things easier, Swift 0.95 merges these different configuration files into a single, common configuration file called swift.properties.

The new swift.properties file is responsible for:

1. Defining sites
2. Defining applications
3. Defining various swift settings

Here is an example of a new swift.properties file.

```
# Define sandyb site
site.sandyb.tasksPerWorker=16
site.sandyb.taskWalltime=00:05:00
site.sandyb.jobManager=slurm
site.sandyb.jobQueue=sandyb
site.sandyb.maxJobs=1
site.sandyb.workdir=/scratch/midway/$USER/work
site.sandyb.filesystem=local

# Define sandyb apps
app.sandyb.echo=/bin/echo

# Define swift properties
sitedir.keep=true
wrapperlog.always.transfer=true

# Select which site to run on
site=sandyb
```

This single swift.properties file works identically to using the sites.xml, tc.data, and cf files listed in the previous section. The details of this file will be explained more later. Let's first look at an example of running Swift with this new file.

Using the swift.properties file above, the new Swift command a user would run is:

```
$ swift script.swift
```

That is all that is needed. Everything Swift needs to know is defined in swift.properties.

## 3.1 Location of swift.properties

Swift searches for swift.properties files in multiple locations:

1. The etc/swift.properties file included with the Swift distribution.
2. $SWIFT_SITE_CONF/swift.properties - used for defining site templates.
3. $HOME/.swift/swift.properties
4. swift.properties in your current directory.
5. Any property file you point to with the command line argument "-properties <file>"

Settings get read in this order. Definitions in the later files will override any previous definitions. For example, if you have execution.retries=10 in $HOME/.swift/swift.properties, and execution.retries=0 in the swift.properties in your current directory, execution.retries will be set to 0.

To verify what files are being read, and what values will be set, run:

```
$ swift -listconfig
```

### 3.2 Selecting a site

There are two ways Swift knows where to run. The first is via swift.properties. The site command specified which site entries should be used for a particular run.

```
site=sandyb
```

Sites can also be selected on the command line by using the -site option.

```
$ swift -site westmere script.swift
```

The -site command line argument will override any sites selected in swift.properties.

### 3.3 Selecting multiple sites

To use multiple sites, use a list of site names separated by commas. In swift.properties:

```
site=westmere,sandyb
```

The same format can be used on the command line:

```
$ swift -site westmere,sandyb script.swift
```

**Note**
You can also use "sites=" in swift.properties, and "-sites x,y,z" on the command line.

### 3.4 Run directories

When you run Swift, you will see a run directory get created. The run directory has the name of runNNN, where NNN starts at 000 and increments for every run.

The run directories can be useful for debugging. They contain: .Run directory contents

| | |
|---|---|
| apps | An apps generated from swift.properties |
| cf | A configuration file generated from swift.properties |
| runNNN.log | The log file generated during the Swift run |
| scriptname-runNNN.d | Debug directory containing wrapper logs |
| scripts | Directory that contains scheduler scripts used for that run |
| sites.xml | A sites.xml generated from swift.properties |
| swift.out | The standard out and standard error generated by Swift |

### 3.5 Using site templates

This new configuration mechanism should make it easier to use site templates. To use this, set the environment variable $SWIFT_SITE_CONF to a directory containing a swift.properties file. This swift.properties can contain multiple site definitions for the various queues available on the cluster you are using.

Your local swift.properties then does not need to define the entire site. It may contain only differences you need to make that are specific to your application, like walltime.

### 3.6 Backward compatability

Swift 0.95 should be backwards compatible with Swift 0.94. If you would like to use XML files and tc.data/app files in the previous style, things should work as before. If you notice an instance where this is not true, please send an email to swift-

support@ci.uchicago.edu.

# 4 The swift.properties file format

## 4.1 Site definitions

Site definitions in the swift.properties files begin with "site". The second word is the name of the site you are defining. In these examples we will define a site called westmere. The third word is the property.

For example:

```
site.westmere.jobQueue=fast
```

Before the site properties are listed, it's important to understand the terminology used.

A **task**, or **app task** is an instance of a program as defined in a Swift app() function.

A **worker** is the program that launches app tasks.

A **job** is related to schedulers. It is the mechanism by which workers are launched.

Below is the list of valid site properties with brief explanations of what they do, and an example swift.properties entry.

Table 1: swift.properties site properties

| **Property** | **Description** | **Example** |
|---|---|---|
| filesystem | Defines how files should be accessed | site.westmere.filesystem=local |
| jobGranularity | Specifies the granularity of a job, in nodes | site.westmere.jobGranularity=2 |
| jobManager | Specifies how jobs will be launched. The supported job managers are "cobalt", "slurm", "condor", "pbs", "lsf", "local", and "sge". | site.westmere.jobManager=slurm |
| jobProject | Set the project name for the job scheduler | site.westmere.project=myproject |
| jobQueue | Set the name of the scheduler queue to use. | site.westmere.jobQueue=westmere |
| jobWalltime | The maximum number amount of time allocated in a scheduler job, in hh:mm:ss format. | site.westmere.jobWalltime=01:00:00 |
| maxJobs | Maximum number of scheduler jobs to submit | site.westmere.maxJobs=20 |
| maxNodesPerJob | The maximum number of nodes to request per scheduler job. | site.westmere.maxNodesPerJob=2 |
| taskDir | Tasks will be run from this directory. In the absence of a taskDir definition, Swift will run the task from workdir. | site.westmere.taskDir=/scratch/local/$USER/wo |
| tasksPerWorker | The number of tasks that each worker can run simultaneously. | site.westmere.tasksPernode=12 |
| taskThrottle | The maximum number of active tasks across all workers. | site.westmere.taskThrottle=100 |
| taskWalltime | The maximum amount of time a task may run, in hh:mm:ss. | site.westmere.taskWalltime=01:00:00 |
| site | Name of site or sites to run on. This is the same as running with swift -site <sitename> | site=westmere |

Table 1: (continued)

| Property | Description | Example |
|---|---|---|
| workdir | The workdirectory element specifies where on the site files can be stored. This directory must be available on all worker nodes that will be used for execution. A shared cluster filesystem is appropriate for this. Note that you need to specify absolute pathname for this field. | site.westmere.workdir=/scratch/midway/$USER/ |

## 4.2 Grouping site properties

The example swift.properties in this document listed the following site related properties:

```
site.westmere.provider=local:slurm
site.westmere.jobsPerNode=12
site.westmere.maxWalltime=00:05:00
site.westmere.queue=westmere
site.westmere.initialScore=10000
site.westmere.filesystem=local
site.westmere.workdir=/scratch/midway/davidkelly999
```

However, it is also simplify this by grouping these properties together with curly brackets.

```
site.westmere {
   provider=local:slurm
   jobsPerNode=12
   maxWalltime=00:05:00
   queue=westmere
   initialScore=10000
   filesystem=local
   workdir=/scratch/midway/$USER/work
}
```

## 4.3 App definitions

In 0.95, applications wildcards will be used by default. This means that $PATH will be searched and pathnames to application do not have to be defined.

In the case where you have multiple sites defined, and you want control over where things run, you will need to define the location of apps. In this scenario, you will can define apps in swift.properties with something like this:

```
app.westmere.cat=/bin/cat
```

When an app is defined in swift.properties for any site you are running on, wildcards will be disabled, and all apps you want to use must be defined.

## 4.4 General Swift properties

Swift properties can be used in the new swift.properties file with no changes. Example:

```
sitedir.keep=true
```

For the list of available properties and their descriptions, please see the User Guide entry for Swift configuration properties.

### 4.5 Using shell variables

Any value in swift.properties may contain environment variables. For example:

```
workdir=/scratch/midway/$USER/work
```

Environment variables are expanded locally on the machine where you are running Swift.

Swift will also define a variable called $RUNDIRECTORY that is the path to the run directory Swift creates. In a case where you'd like your work directory to be in the runNNN directory, you may do something like this:

```
workdir=$RUNDIRECTORY
```