Swift 0.94 Release Notes

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Iterate changes

To behavior of iterate has changed from 0.93 to 0.94. The until() clause of the iterate statement now causes the loop to stop as soon as the until clause returns "true", instead of doing one additional iteration when the clause turns true.

For example, iterate code in 0.93 might say:

```
iterate i {
...
} until (i == 9);
```

To achieve the same 10 executions in 0.94, you must say:

```
iterate i {
...
} until (i==10);
```

More information about iterate can be found in the iterate section of the Swift User Guide.

2 Coaster walltimes

Walltimes are now strictly enforced by coasters.

There are two coaster properties that affect how walltimes are handled. They are maxtime and maxwalltime.

Maxtime is the maximum amount of time, in seconds, that a coaster block can run. This is defined in sites.xml.

Setting a maxTime of 30 minutes

```
<profile namespace="globus" key="maxTime">1800</profile>
```

The maxwalltime property defines the maximum amount of time that any given application may run (in hh:mm:ss format)

Setting a maxWallTime of 5 minutes

```
<profile namespace="globus" key="maxWallTime">00:05:00</profile>
```

An application may run up to the the limit defined in maxwalltime, and are packed into coasters slots that run for maxtime seconds.

In versions of Swift prior to 0.94, if an application exceeded its specified maxwalltime, execution would continue. However, if the application exceeded its maxwalltime, the queuing system could kill the worker. The resulting error message was not always very clear.

In Swift 0.94, coaster workers enforce the user-specified maxwalltime. If an application exceeds its maxwalltime, the coaster worker will not allow it to continue, but terminate it and report the error.

The error message will look something like this:

```
Execution failed:
        Exception in hostsnsleep:
    Arguments: [30]
    Host: midway
    Directory: hostsnsleep-20130502-2009-t8m0l3hb/jobs/n/hostsnsleep-n55v8x8l
    stderr.txt:
    stdout.txt: midway009
Caused by:
        Walltime exceeded
org.globus.cog.abstraction.impl.common.execution.JobException: Walltime exceeded (exit code ↩
    : 513)
        at org.globus.cog.abstraction.coaster.service.local.JobStatusHandler. ↩
            requestComplete(JobStatusHandler.java:38)
```

```
        at org.globus.cog.karajan.workflow.service.handlers.RequestHandler.receiveCompleted ↩
            (RequestHandler.java:88)
        at org.globus.cog.karajan.workflow.service.channels.AbstractKarajanChannel. ↩
            handleRequest(AbstractKarajanChannel.java:518)
        at org.globus.cog.karajan.workflow.service.channels.AbstractStreamKarajanChannel. ↩
            stepNIO(AbstractStreamKarajanChannel.java:238)
        at org.globus.cog.karajan.workflow.service.channels.NIOMultiplexer.loop( ↩
            NIOMultiplexer.java:97)
        at org.globus.cog.karajan.workflow.service.channels.NIOMultiplexer.run( ↩
            NIOMultiplexer.java:56)
(exit code: 513)
        hostsnsleep, hostsnsleep.swift, line 13
```

Tip
Maxwalltime may also be define on a per application level in your tc file (Globus Namespace section, Swift User Guide).

3 Camel Case Functions

To make things more consistent, many swift functions are being renamed to use camel case naming. For example, @toint is now @toInt. The previous naming conventions will still work in 0.94, but may not in future releases. If you try to use a function which has been renamed, you will see deprecated warnings like this:

```
Warning: Function toint is deprecated, at 2
```

Below is a list of functions with updated names.

Old Name	New Name
@extractint	@extractInt
@toint	@toInt
@tofloat	@toFloat
@tostring	@toString

4 Associative Arrays

Associative arrays have been added. By default, array keys are integers, but in 0.94 other primitive types are also allowed as array keys. More details and examples can be found in the associate array section of the Swift User Guide.

5 Dynamic profiles

Many settings formerly only definable in sites.xml can now be set on a per-app basis. This can make things easier when running multiple apps that have different requirements for settings like processors per node and walltime.

More information and examples can be found in the dynamic profiles section of the Swift User Guide

6 SSH command line provider

Added a new ssh command line provider. Previously ssh support was done by creating a file called ~/.ssh/auth.defaults. The ssh command line provider is more flexible and doesn't require this step. ssh-cl allows you to use your existing SSH agents. You can use ssh-cl to run remotely by adding something like this to your sites.xml:

```
<execution provider="coaster" url="my.host.uchicago.edu" jobmanager="ssh-cl:pbs"/>
```

7 Coaster provider staging improvements

Coasters provides a mechanism for transferring files between hosts, called coaster provider staging. Coaster provider staging is explained in more detail in this research paper.

In 0.94 there have been many fixes and improvements to make coaster provider staging more reliable.

8 Added support for the Slurm scheduler

Swift 0.94 includes support for Slurm - an open source job scheduler used on many supercomputers and computer clusters.

The Swift Site Guide entry for Midway has an entry containing information about how to run Swift on a specific Slurm cluster at the University of Chicago called Midway. By changing queue names and project names, this information could be used to run on any Slurm cluster.

9 Added support for the LSF scheduler

Swift 0.94 includes support for the LSF (Load Sharing Facility) scheduler. An example sites.xml can be found in the Swift Site Guide entry for Geyser.

10 Condor Improvements

Some improvements have been made to the condor provider. For condor systems without a shared filesystem, Swift can now use Condor's built-in file transfer mechanisms to transfer the scripts it needs to starts jobs. To do this, you can define the jobtype to "non-shared" in sites.xml.

```
<profile namespace="globus"  key="jobType">nonshared</profile>
```

An example setup that uses this style of configuration can be found in the Swift siteguide entry for the UC3 cluster.

11 Textual User Interface

Fixes for the textual user interface (TUI). Adding the -tui option to the swift command line allows you to monitor progress in a curses based menu. For example:

```
swift -sites.file sites.xml -tc.file tc.data -tui myscript.swift
```

A brief example of this can be found at http://www.ci.uchicago.edu/~davidk/modis.ogv.

12 @java

Added the ability to call Java methods within swift using @java. For example:

```
float f = @java("java.lang.Math", "sin", 0.5);
```

More information can be found in the Swift User Guide entry on @java.

13 Hang Checker

Added a hang checker that provides the user with more information about potential hangs. Here is one example of a situation that may cause Swift to hang:

```
int a;
int b;

a = b;
b = a;
```

When this script is run, you will now see a more detailed message explaining what's happening.

```
Dependency loop found:
        b (declared on line 2) is needed by:
                a = b, test.swift, line 5
        which produces a (declared on line 1)

        a (declared on line 1) is needed by:
                b = a, test.swift, line 6
        which produces b (declared on line 2)
```

14 @strjoin

A function called @strjoin function for joining strings. More information and examples are in the Swift User Guide section on @strjoin.

15 Worker Init Command

If you have a requirement that a command get run on the worker node before processing any work, worker.pl will now execute commands stored in the environment variable $WORKER_INIT_CMD.

```
export WORKER_INIT_CMD=/path/to/script.to.run
```

16 Swift Userhome

Swift creates some files it needs by default in your home directory. If your home filesystem is not available on the worker nodes of a cluster you are trying to use, this can cause tasks to fail. Setting the environment variable SWIFT_USERHOME to the location of a shared filesystem will get around this restriction.

17 MPI Improvements

Swift 0.94 has improved MPI support and integration. Details about how to run MPI jobs in Swift can be found in the the MPI section of the Swift User Guide.