Pipeline II Variables and Substreams

Pipeline II Variables and Substreams

Pipeline II introduces a new concept of variables. Variables can be defined in several places:

1. In the XML file which defines a task, at the level of a task, a process.
2. When a stream or substream is created or rolled back
3. By script processes ("scriptlets")
4. By batch jobs

Variables specified in the XML are typically used to define "constants", while variables passed in to the stream or computed by processes provide the ability to perform runtime calculations.

Defining constants in XML

Variables can be defined within the <task> or <process> tag.

example.xml

```
<variables>
   <var name="configdir">${jobdirroot}/config</var>
   <var name="mcSrcIdfile">${outdir}/${pipeline.task}-${sixdigstream}_mcsrcId.txt</var>
   <var name="jobdirroot">/nfs/farm/g/glast/u26/MC-tasks/${pipeline.task}/</var>
   <var name="outdir">${jobdirroot}/output/${sixdigstream}</var>
   <var name="scDatafileLD">${LDprefix}/scData:run-${sixdigstream}</var>
   <var name="mcSrcIdfileLD">${LDprefix}/mcSrcId:run-${sixdigstream}</var>
   <var name="scDatafile">${outdir}/${pipeline.task}-${sixdigstream}_scData_0000.fits</var>
   <var name="eventfile">${outdir}/${pipeline.task}-${sixdigstream}_events_0000.fits</var>
   <var name="LDprefix">/ServiceChallenge/${pipeline.task}</var>
   <var name="sixdigstream">${format(pipeline.stream,"%06d")}</var>
   <var name="eventfileLD">${LDprefix}/events:run-${sixdigstream}</var>
</variables>
```

Defining constants at stream creation

There are currently three ways to create new streams

1. Using the pipeline II web interface
2. Using the pipeline II command line interface
3. Using the pipeline II Java client

In the first two cases variables are passed in as a string, of the form:

```
<variable>=<value>[,<variable>=<value>...]
```

for example:

```
a=3,b=2.5,c="Something"
```

The type assigned to the variable is based on its value, for example in the above example a is an integer, b is a float and c is a string.

In the third case (using the Java client) variables can be specified either as a string, or as a Map<String,Object> of variables names and values.

Prerequisites

Prerequisites allow the XML file that defines a task to require that certain variables be set at stream creation time. If values are not given the stream creation will fail. Prerequisites are just the minimal set of variables that must be set, it is always possible to specify additional variables.

Defining constants at stream rollback time

Currently stream rollback is only available via the web interface. When rolling back streams you will be given an opportunity to give new values to any variables set at the stream level, or to define new variables.

Defining variables in a batch job

When a batch job is run by the pipeline it creates a file called pipeline_summary in the working directory of the job. This file is sent (by email) to the pipeline server at the start and end of the batch job. The file is in the format of a Java properties file. It contains by default information required by the pipeline itself (see example below), but users can add extra lines of the form:

```
Pipeline.<variable>: <value>
```

Any such lines will be recorded in the pipeline database by the pipeline server, and will become variables accessible to other processes running in the same stream.

Bash functions have been set up to make it slightly easier to do this from a bash script:

```
pipelineSet <variable> <value>
pipelineCreateStream <subtask> <stream> [<env-var-list>]
```

Don't use pipeline_summary filename directly

Don't rely on the summary file being called pipeline_summary, as this may change in future. Rather use the environment variable PIPELINE_SUMMARY which will translate to the absolute path of the summary file.

Example pipeline_summary file

```
From: pipeline-prod@slac.stanford.edu
To: pipeline@slac.stanford.edu
Subject: 11513
Errors-To: tonyj@slac.stanford.edu

ProcessInstance: 11513
Host: cob0343
StartTime: Sun Nov 19 20:02:47 PST 2006
WorkDir: /nfs/farm/g/glast/u26/MC-tasks/obssim-ST-v7r6p1/output/000001
LogFile: /nfs/farm/g/glast/u26/MC-tasks/obssim-ST-v7r6p1/output/000001/logFile.txt
Elapsed: 5231.17
User: 5151.14
System: 4.19
ExitCode: 0
EndTime: Sun Nov 19 21:29:59 PST 2006
Signals:
Status: Ended
```

Defining variables in a scriplet

Pipeline tasks can contain scriptlets, which must currently by written in Jython. Each scriptlet has a pre-defined object called pipeline which can be used to interact with other processes in the same stream or within a substream of the current stream. The pipeline object contains one method which allows variables to be defined by the scriptlet.

```
void setVariable(String varName, Object value);
```

Any such lines will be recorded in the pipeline database by the pipeline server, and will become variables accessible to other processes running in the same stream.

Accessing variables from a scriptlet

Before a scriptlet executes the pipeline server collects all the variables from

1. The task
2. The process
3. Any parent streams
4. The current stream

and makes all of these available as predefined (python) variables in the scriplet. They can be used just like any other language variable in the scriptlet.

In addition the scriplet has access to a number of methods which allow it to access variables from other processes in the same stream, or substream.

Pipeline.java

```
ProcessInstance getProcessInstance(String process);
Substream getSubstream(String subtask, int stream);
List<Substream> getSubstreams(String subtask);
```

ProcessInstance.java

```
String getName();
Date getEndDate();
Date getStartDate();
Date getSubmitDate();
int getStream();
String getStatus();
int getExecutionNumber();
int getExitCode();
Object getVariable(String varName);
Map<String, Object> getVariables();
```

Substream.java

```
int getStream();
String getTask();
ProcessInstance getProcessInstance(String process);
```

See the Java Documentation for up-to-date details on these classes.

Accessing variables from a batch job

Before a batch job executes the pipeline server collects all the variables from

1. The task
2. The process
3. Any parent streams
4. The current stream

and makes all of these available as environment variables in the batch job.

To Do

Currently all of the variables defined above are made available as environment variables, in future we may make it possible to control which variables become environment variables.

If it is required for a batch job to access a variable from a previous process this can be achieved by explicitly setting a variable in the process which defines the job.

ExampleJob.xml

```
<process name="stepB">
   <notation>Scriptlet Test Step B</notation>
   <variables>
       <var name="DERIVED">Stream ${pipeline.stream} ${pipeline.getProcessInstance("stepA").getVariable
("message")}</var>
   </variables>
   <job>
       pipeline-set message "Hello from batch: ${DERIVED}"
   </job>
   <depends>
       <after process="stepA"/>
   </depends>
</process>
```

A complete example

The following example shows many of the features discussed above.

```
<?xml version="1.0" encoding="UTF-8"?>
<pipeline
   xmlns="http://glast-ground.slac.stanford.edu/pipeline"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://glast-ground.slac.stanford.edu/pipeline http://glast-ground.slac.stanford.edu
/Pipeline-II/schemas/2.0/pipeline.xsd">
   <task name="ScriptletTest" version="2.2" type="Data">
      <notation>Just Testing</notation>
      <process name="step1">
         <notation>Scriptlet Test Step 1</notation>
         <!--
         <script language="python">
            <![CDATA[
               pipeline.setVariable("message","Hello from Step1")
               for i in range(10,12):
                  pipeline.createSubstream("TestSubtask",i)
```