IBM Spectrum LSF Process Manager 10.2

Using Templates with the IBM Spectrum LSF Process Manager Clients

Tables of Contents

Working with templates

IBM Spectrum LSF (LSF) templates extend the functionality of LSF

Templates allow you to accomplish the following tasks:

- Use LSF to submit work to other applications outside of LSF without intensive programming
- Create a customized user interface for LSF users by pre-configuring the execution environment
  LSF templates are XML documents that define the properties of a job and the options required to configure and run the job. A properly coded LSF template produces a property editor, which is a simplistic user interface that allows the user to visually configure the job. The Flow Editor constructs the property editor from the Job Template Definition file at runtime.

  The Job Template Definition follows well-known XML schema. Each definition is registered with the Flow Editor, at which point it can be used as part of a flow. The user configures the job by setting properties and options accessed via the property editor. All information entered into the property editor by the user is kept as part of the flow definition, and is submitted to LSF for execution.

Template location

LSF templates must be stored in the templates directory before they can be recognized by Process Manager. When a template is added to the directory, it appears in the list of templates available to the user. Ensure that any completed templates are stored in *JS_HOME*/work/templates.
Note: If the template directory is to be moved to a different location, the LSF configuration file (js.conf) needs to contain a new parameter, JS_TEMPLATE_DIR, and LSF needs to be reconfigured.

- **Job Template DTD**

Job Template DTD

The following is the DTD (Document Type Definition) for the job template:

```
<!ELEMENT JobTemplate (Execution | Submission | Help)*>
<!ELEMENT Execution (Param)*>
<!ELEMENT Submission (Param)*>
<!ELEMENT Param (Choice | Help)*>
<!ELEMENT Choice EMPTY>
<!ELEMENT Help EMPTY>

<!ATTLIST JobTemplate Name CDATA #REQUIRED>
<!ATTLIST JobTemplate Version CDATA #IMPLIED>
<!ATTLIST JobTemplate Icon CDATA #IMPLIED>

<!ATTLIST Execution Type (lsf | oracle | zos) "lsf">
<!ATTLIST Execution Action CDATA #REQUIRED>

<!ATTLIST Param Name CDATA #REQUIRED>
<!ATTLIST Param Expr CDATA #IMPLIED>
<!ATTLIST Param Caption CDATA #IMPLIED>
<!ATTLIST Param Type (STRING|INTEGER|FLOAT|FILE|FOLDER|PASSWORD) "STRING">
```

```
<!ATTLIST Param Default CDATA #IMPLIED>
<!ATTLIST Param Repeatable (True | False) "False">
<!ATTLIST Param Required (True | False) "True">
<!ATTLIST Param Readonly (True | False) "False">
<!ATTLIST Param Hidden (True | False) "False">
<!ATTLIST Param Lookup (True | False) "False">

<!ATTLIST Choice Name CDATA #IMPLIED>
<!ATTLIST Choice Value CDATA #REQUIRED>
```

Note:
While most values in the DTD are case-sensitive, you can specify True and False in uppercase, lowercase, or mixed case, as shown in this document.

Template Components and Format

This section describes the elements that make up the properly constructed XML template document and the attributes and child elements for each element.

JobTemplate element

The `JobTemplate` element defines this document as a job template. There can be only one of job template defined in the XML file. The template is defined in the contents between the <**JobTemplate**> and </`JobTemplate`> tags.

Attributes

The `JobTemplate` element has the following attributes:

- `Name`
- `Version`
- `Icon`

Name="*template_name*"
: Specifies the name of the job template. This is the name that is displayed in the list of applications to choose from in the Flow Editor.

  Specify alphanumeric characters or underscore, and enclose the name in single or double quotes.

Version="*version*"
: Specifies the version of the job template. Use this attribute to help you track changes to the template.

  Specify alphanumeric characters, or period, comma or underscore, and enclose the version in single or double quotes.

Icon="*file_name*"
: Specifies the file name of an image to be used as an icon to represent this type of job in the flow. This file must be located in JS_HOME/work/templates, or in the template directory as specified in the LSF configuration file, js.conf.

Specify a non-transparent GIF file 32 x 32 pixels and 256 colors.

Child elements

The `JobTemplate` element has the following child elements:

- `Execution`
- `Submission`
- `Help`

Execution element

The `Execution` element defines a block of execution environment configuration parameters. The execution environment is defined in the contents between the <**Execution**> and </`Execution`> tags.

Attributes

The `Execution` element has the following attributes:

- `Type`
- `Action`

Type="*type*"
: Specifies the type of job execution. Currently, the valid values for `Type` are `lsf`, `oracle`, and `zos`.

Action="*action*"
: Specifies the executable or internal shell command that runs the job. For example:

"c:\tmp\myexe.exe"

or

"ls"

Child elements

The `Execution` element has the following child element:

- `Param`

Submission element

The `Submission` element defines one or more parameters that are passed directly to the executable or command specified in `Execution.Action`. The `Submission` element has no attributes.

Child elements

The `Submission` element has the following child element:

- `Param`

Param element

The `Param` element has the following attributes:

- `Name`
- `Caption`
- `Type`
- `Expr`
- `Required`
- `Default`
- `Repeatable`
- Hidden
- Lookup
- Readonly

Name="*parameter_name*"
: Specifies the name of the parameter.

  Specify alphanumeric characters or underscore, and ensure the name is unique within the template.

Caption="*caption*"
: Specifies the text that will appear in the user interface for this parameter. If you do not specify a value, the text specified for `Name` is used.

Type="STRING|INTEGER|FILE|FLOAT|FOLDER|PASSWORD"
: Specifies the validation rule to apply to user input for this parameter. The values are as follows:

Value	Description
STRING	The user must enter a text string. This is the default rule.
INTEGER	The user must enter an integer. The range of allowed values will be enforced by the Java™ runtime
FILE	The user must enter a file name.
FLOAT	The user must enter a number, which may contain a decimal point. The range of allowed values will be enforced by the Java™ runtime.
FOLDER	The user must enter a directory name.
PASSWORD	The user input is a password. The text is not displayed in the property editor when the user types it.

Expr="*expression*"
: Specifies the expression that is used to format and encode the user input for this parameter. This expression can contain an internal variable. The variable is replaced with the text the user typed when the template is evaluated. For example,

  `"-d #{VALUE}"`

  The default value is `"#{VALUE}"`, the value as entered by the user.

Default="*default*"

Specifies the default value for this parameter, if one exists. The text provided here is subject to parameter validation.

Required="TRUE | FALSE"
: Specifies if this parameter is required. The default is "TRUE".

Repeatable="TRUE | FALSE"
: Specifies if this parameter can be specified more than once on the command line. The default is FALSE.

Readonly="TRUE | FALSE"
: Specifies if the user can change the value. Specify `Readonly="TRUE"` to prevent the user from changing the value. Specify `Readonly="FALSE"` to allow the user to change the value. The default is FALSE.

Lookup="TRUE | FALSE"
: Specifies if the user can also type directly in the field. Specify `Lookup="TRUE"` to force the user to select from the list. Specify `Lookup="FALSE"` to allow the user to select from the list or type directly in the input field. The default is FALSE.

Hidden="TRUE | FALSE"
: Specifies if this parameter is exposed to the user. The default is FALSE.

Child elements

The `Param` element has the following child element:

- `Choice`
- `Help`

Choice element

The `Choice` element presents a valid value for the parameter. A parameter may have multiple choices.

Attributes

The `Choice` element has the following attributes:

- `Caption`
- `Value`

Caption="*caption*"
: Specifies the text that will appear in the user interface for this choice. If no value is specified, the caption is derived from the value.

Value="*value*"
: Specifies the text that is used by the parameter expression if the user selects this choice.

Help element

The `Help` element defines a help string within the context of another element. The `Help` element is contained between the <`Help`> and </`Help`> tags. The `Help` element has no attributes.

Using the z/OS® Template

The z/OS template allows Process Manager users to schedule mainframe jobs from within their Process Manager flows. The z/OS template is ready for use; simply ensure that `zos_Template.xml` is located in the *JS_HOME*/work/templates directory.

About the z/OS jobs

Process Manager uses the job you define to submit an LSF® job (referred to here as a proxy job), which in turn file transfers JCL to the mainframe to run. The job output is captured and file transferred to an output file location you specify.

Use the z/OS template

- Ensure `zos_Template.xml` is installed in the user's templates directory *JS_HOME*/work/templates. You can define a z/OS job from the Flow Editor or from the command line.

- **Define a z/OS job from the Flow Editor**
- **Exit codes**
- **Create a template**

Define a z/OS job from the Flow Editor

Procedure

1. Click the application icon to put the Flow Editor into application placement mode and select z/OS job. The application icon appears in the workspace.
2. Double-click on the application icon to define the job. The Application Definition- Edit Application display appears.
3. On the General tab, fill out the following fields:
    a. In the Name field, specify a unique, meaningful name for the job in the flow.

    b. Optional: In the Description field, add any descriptive text that may be used for managing this job within the flow.

    c. In the z/OS host name field, specify the name of the mainframe host to run the job on.

    d. In the Login user ID and Password fields respectively, specify a valid user ID and password under which to log into the mainframe host to run the job.

    e. In the JCL file field, specify the full path to the JCL to run. This file will be sent to the mainframe host.

    f. In the Output file field, specify the full path to the output file to receive the z/OS® job output.

    g. Optional: In the Estimated run time (in minutes) field, you may specify the approximate number of minutes the job is expected to take to run. The default is 1 minute.

h. Optional: In the Check interval (in minutes) field, you may specify the number of minutes to wait in between checking for the job output. The default is 1 minute.

i. Optional: In the Time out (in minutes) field, you may specify the number of minutes to wait before assuming the job failed. The default is 0, or no timeout period.

4. Optional: On the Execution Environment tab, you may fill out one or more of the following fields:
   a. If you want to submit your job to a particular queue, in the Submit to queue/partition field, specify the queue name. Ensure that the JCL is accessible from any host in the queue you specify.

   b. If you want to run the proxy LSF® job on a specific host, in the Run on host field, specify the name of the host. Ensure that the JCL is accessible from the host you specify.

5. When you have completed filling in the fields, click OK. Draw any dependencies as required.

Results

The job output is in the following format:

- The first column displays the job name.
- The second column displays the job ID
- The third column displays the status of the job, where valid status is OUTPUT. Additionally, the status displays the number of files spooled for output.

  If no output is returned from the job, the job is presumed to have failed.

Required fields

When defining a z/OS job, the user is required to enter the following fields:

- z/OS host name—the name of the mainframe host to log into
- login user ID—the user ID under which to log into the mainframe host
- Password—the password used to validate the login user ID on the mainframe host
- The full path to the JCL to submit
- The full path to the output file to receive the z/OS job output
- (a single spool file—any existing file will be overwritten)

  The JCL and output file names must be accessible to the LSF execution host

Exit codes

If the proxy LSF® job fails, you can determine the location of the failure by the exit code specified in the job details, as follows:

The following exit codes may occur if there is a problem between your LSF proxy job and the mainframe job:

Exit Code	Failure Reason
230	Failed to connect to mainframe via FTP.
231	Failed to log in to mainframe.
232	Command *site filetype=jes* or *site filetype=seq* failed.
233	Failed to retrieve job ID. *put* or *get* command failed.
234	Failed to retrieve job output.
235	Failed to match Dir Header. *Dir* command failed.
236	Failed to get job status. *Dir* command failed.
237	Timeout checking mainframe job status.
238	Failed to delete a mainframe job.
239	Encryption error.
240	Environment variable not found.
241	Script error.
242	LSF configuration file not found.
243	FTP configuration file not found.
244	Incomplete system output file: IEF142I and IEF472I not found.
245	System output file not found.
246	*bpost* command failed.
247	*bread* command failed.
248	Failed to get mainframe job ID from *bread* output.
249	Failed to transfer JCL file from z/OS® host to LSF host.
250	Failed to modify job name in JCL file.
255	Mainframe job has an ABEND status.

A template file should contain all of the information needed by the end user to define a job for use in a specific application. The template you create should use simple, recognizable terms, and can contain help for some or all fields. You can use the template to highlight only those fields where user input is critical, and allow all others to use the default values. This can be useful for simplifying complex submission options.

This procedure demonstrates how to take a job submission command for an application and translate it into a template using the following command, which submits a blast job that searches a database for matches:

```
blastall.exe -p blastp -d "human" -e 11.02 -o "c:\tmp\zzz.txt" -F T -S 2 -T F -1 -U F
```

In the above example, `blastall.exe` is the executable, and the remaining values are parameters that describe how to run the executable: `-p blastp` specifies the BLAST search to run, `-d human` specifies the database to search, and so on.

Create a template

Procedure

1. Using an XML or text editor, start a job template with a meaningful name. Specify the `Version` and `Icon` attributes, if applicable. For example:

```
<JobTemplate Nameplates search" Version="1" Icon="ncbi.jpg">
</JobTemplate>
```

2. Specify the `Execution` element with LSF® as the execution environment (`Type=lsf`), and specify the name of the executable to use. For example:

```
<JobTemplate Name="BLAST search" Version="1" Icon="ncbi.jpg">
   <Execution Type="lsf" Action="blastall.exe"/>
</JobTemplate>
```

   The template type of lsf submits an LSF job using the LSF **bsub** command. If you are creating a template that submits an LSF job, specify the name of the command to run for Action. For example:

```
<Execution Type="lsf" Action="sleep 5"/>
```

3. Specify an empty `Submission` element, which will contain all of the parameters required by the executable. For example:

```
<JobTemplate Name="BLAST search" Version="1" Icon="ncbi.jpg">
   <Execution Type="lsf" Action="blastall.exe"/>
   <Submission>
   </Submission>
</JobTemplate>
```

4. For each parameter in the command, create a corresponding `Param` element. For each `Param` element, specify a `Name` and `Validation` attribute. Specify an `Expr` attribute, if applicable. If the parameter is not required, specify the `Required` attribute and set it to false. Insert the parameters into the `Submission` element. For example:

```
<JobTemplate Name="BLAST search" Version="1" Icon="ncbi.jpg">
   <Execution Type="lsf" Action="blastall.exe"/>
   <Submission>
      <Param Name="Program name" Validation="CHOICE" Expr=" -p #{VALUE}>
         <Choice Name="blastp search" Value="blastp"/>
         <Choice Name="blastn search" Value="blastn"/>
         <Choice Name="blastx search" Value="blastx"/>
      </Param>
      <Param Name="Database" Validation="FILE" Expr=" -d #{VALUE}>
      </Param>
   </Submission>
</JobTemplate>
```

5. Refer to "LSF execution type", and create any execution parameters required as applicable. You can allow many of the execution parameters to default, or you can set a value for a parameter and mark it as hidden or read only. Add these parameters as `Param` child elements to the `Execution` element. For example:

```
<JobTemplate Name="BLAST search" Version="1" Icon="ncbi.jpg">
<Execution Type="lsf" Action="blastall.exe">
     <Param Name="Stdin" Caption="Standard in:" Default="inputfile.txt"
Readonly="True"/>
        <Param Name="Stdout" Caption="Standard out:" Default="outfile.txt"
Readonly="TRUE"/>
        <Param Name="JobPriority" Default="standard" Hidden="True"/>
  </Execution>
  <Submission>
     <Param Name="Program name" Validation="CHOICE" Expr=" -p #{VALUE}>
        <Choice Name="blastp search" Value="blastp"/>
        <Choice Name="blastn search" Value="blastn"/>
        <Choice Name="blastx search" Value="blastx"/>
     </Param>
     <Param Name="Database" Validation="FILE" Expr=" -d #{VALUE}/>
```

```
  </Submission>
</JobTemplate>
```

6. Save the file with a meaningful name with a file type of `.xml` in the following location: *JS_HOME*/work/templates or in the template directory specified in the Process Manager configuration file, js.conf.

Example Templates

The Process Manager package includes sample templates you can use, either as is or modified to fit your environment. The package contains the following examples:

- Example_Oracle_Template.xml—runs an Oracle procedure
- zOS _Template.xml—runs a z/OS® job and retrieves the job output

  You can find these examples in *JS_HOME*/10.2/examples.

Execution Types

Execution type determines the kind of work that is run and defines the execution environment. The following are the valid execution types supported by LSF:

- lsf
- zos
- oracle

Each execution type has a predefined list of execution parameters it supports. Select the execution parameters that apply. Be sure to enter the exact parameter name as shown in the list that follows.

Reserved parameter names

Execution parameters are predefined. When you specify a parameter with the same name as any of the predefined parameters, your template uses the predefined functionality. Therefore you cannot create a parameter with the same name as any of the execution parameters to perform another function.

LSF execution type

The LSF® execution type is used to run LSF jobs.

Valid execution parameters:

All execution parameters listed are valid in an LSF template.

z/OS execution type

The zos execution type is used to run mainframe jobs running on z/OS®.

Valid execution parameters:

The zos execution type supports only the following execution parameters:

- FileTransfer
- HostSelect
- QueueSelect
- UserName

Oracle execution type

The Oracle execution type is used to run Oracle procedures.

Valid execution parameters:

All execution parameters listed are valid in an Oracle template.

Execution Parameters

Execution parameters define the execution environment for an application. These are the parameters that are used by LSF directly. The following are predefined parameters you can use in your templates.

The execution parameters that are valid for a given template type are predetermined—be sure you refer to “Execution Types” to see which parameters are valid for the execution type you are using.

CoreSizeLimit

Specifies the maximum core file size limit (in KB) for any process that belongs to this batch job.

Default format

```
<Param Name=”CoreSizeLimit” Type=”FLOAT” Expr=”#{VALUE}” Required=”TRUE”
Repeatable=”FALSE” Hidden=”FALSE”/>
```

Example

```
<Param Name=”CoreSizeLimit” Caption=”Max. core size (KB):”/>
```

CPUTimeLimit

Specifies the maximum amount of CPU time the job can use. The limit is specified as *hours*:*minutes* or just *minutes*.

Default format

```
<Param Name="CPUTimeLimit” Type=”STRING” Expr=”#{VALUE}” Required=”TRUE”
Repeatable=”FALSE” Hidden=”FALSE”/>
```

Example

```
<Param Name=”CPUTimeLimit” Caption=”Max. CPU time (minutes):”/>
```

DataSizeLimit

Specifies the maximum data segment size (in KB) for any process that belongs to this job.

Default format

```
<Param Name="DataSizeLimit" Type="FLOAT" Expr="#{VALUE}" Required="TRUE"
Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="DataSizeLimit" Caption="Max. data size (KB):"/>
```

ExclusiveExec

Specifies that this job requires exclusive use of the host while it is running. The user specifies Yes or No.

Default format

```
<Param Name="ExclusiveExec" Type="STRING" Expr="#{VALUE}" Required="TRUE"
Repeatable="FALSE" Hidden="FALSE" Default="No" Lookup="TRUE">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

Example

```
<Param Name="ExclusiveExec" Caption="Exclusive use of host:"
Default="No">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

FileSizeLimit

Specifies the maximum file size limit (in KB) for any process that belongs to this job.

Default format

```
<Param Name="FileSizeLimit" Type="FLOAT" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="FileSizeLimit" Caption="Max. file size (KB):"/>
```

FileTransfer

Specifies one or more file transfer operations required to run this job. This option can be used to transfer local files to the remote host for processing, to return output files from the remote host after running the job, or to append output files to a local location after running the job.

Default format

```
<Param Name="FileTransfer" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="FileTransfer" Caption="Transfer files:"/>
```

Hold

Specifies to submit the job on hold. The job cannot run until it is explicitly released. The user specifies Yes or No.

Default format

```
<Param Name="Hold" Type="STRING" Expr="#{VALUE}"
 Required="FALSE" Repeatable="FALSE" Hidden="FALSE" Default="No"
 Lookup="TRUE">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

Example

```
<Param Name="Hold" Caption="Submit on hold:" Default="No">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

HostDependent

Specifies that this job must run on the same host as a previous job in the flow. The user input is the name of the job. The previous job must have at least started to run when this job is submitted, so the correct host can be determined.

Default format

```
<Param Name="HostDependent" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="HostDependent" Caption="Run on same host as:"/>
```

HostSelect

Specifies that this job must run on one of the specified host or series of hosts. The user input is the name of the host.

Default format

```
<Param Name="HostSelect" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="HostSelect" Caption="Run on hosts:"/>
```

JobPriority

Specifies a priority to assign this job. This allows users to order their jobs in the queue. Jobs are still subject to all scheduling policies regardless of job priority. Jobs with the same priority are ordered first come first served. The user input is an integer between 1 and the maximum configured user priority.

Default format

```
<Param Name="JobPriority" Type="INTEGER" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="JobPriority" Caption="Priority"
Type="INTEGER:"/>
```

LoginShell

Specifies the login shell to use to initialize the execution environment. This is not necessarily the shell under which the job runs. The user input is the absolute path to the login shell.

Default format

```
<Param Name="LoginShell" Type="FILE" Expr="#{VALUE}"
 Required="FALSE" Repeatable="TRUE" Hidden="FALSE"
 Default="korn"/>
```

Example

```
<Param Name="LoginShell" Caption="Login shell:"/>
```

MailDestination

Specifies the e-mail address to notify regarding the state of the job. The user input is the e-mail address.

Default format

```
<Param Name="MailDestination" Type="STRING" Expr="#{VALUE}"
 Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="MailDestination" Caption="Email address:"/>
```

MailWhenFinish

Specifies to send an e-mail notification when the job finishes. The user specifies Yes or No.

Default format

```
<Param Name="MailWhenFinish" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE" Default="No"
Lookup="TRUE">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

Example

```
<Param Name="MailWhenFinish" Caption="Notify when finishes:"
 Default="No">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

MailWhenStart

Specifies to send an email notification when the job starts. The user specifies Yes or No.

Default format

```
<Param Name="MailWhenStart" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE" Default="No"
 Lookup="TRUE">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

Example

```
<Param Name="MailWhenStart" Caption="Notify when starts:"
 Default="No">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

MinMaxCPU

Specifies the minimum and maximum number of processors this parallel job can use. The maximum number is optional—if the user specifies only a minimum number, that is the actual number used. The user input is string `minimum,maximum` or just `minimum`.

Default format

```
<Param Name="MinMaxCPU" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="MinMaxCPU" Caption="# Processors:"/>
```

PeakMemLimit

Specifies the maximum amount of memory this job can use, in kilobytes. The user input is an integer.

Default format

```
<Param Name="PeakMemLimit" Type="FLOAT" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="PeakMemLimit" Caption="Max. memory (KB):" Type="INTEGER"/>
```

PeakSwapLimit

Specifies the maximum amount of swap space this job can use, in kilobytes. The user input is the number of kilobytes.

Default format

```
<Param Name="PeakSwapLimit" Type="FLOAT" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="TRUE"/>
```

Example

```
<Param Name="PeakSwapLimit" Caption="Max. swap (KB):"
Type="INTEGER"/>
```

ProjectName

Specifies the project code for collecting accounting information. The user input is the project name.

Default format

```
<Param Name="ProjectName" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="ProjectName" Caption="Project:"/>
```

QueueSelect

Specifies the queue to which this job should be submitted. All jobs submitted to a queue share the same scheduling and control policies. The user input is the name of the queue, or you can provide a list of defined queues for your site.

Default format

```
<Param Name="QueueSelect" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="QueueSelect" Caption="Queue:"
Default="general" Lookup="FALSE">
   <Choice Name="Priority" Value="priority"/>
   <Choice Name="General" Value="general"/>
</Param>
```

Rerunable

Specifies that this job is rerunnable—if the host becomes unavailable, the job can be restarted on another host. The user specifies Yes or No.

Default format

```
<Param Name="Rerunable" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE" Default="No"
Lookup="TRUE">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

Example

```
<Param Name="Rerunable" Caption="Rerunnable:"
Default="Yes">
   <Choice Name="True" Value="Yes"/>
   <Choice Name="False" Value="No"/>
</Param>
```

ResReqStr

Specifies the resources required to run this job. The user input is a string.

Default format

```
<Param Name="ResReqStr" Type="STRING" Expr="#{VALUE}"
 Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="ResReqStr" Caption="Resources required:"/>
```

RunTimeLimit

Specifies the maximum run time of the job in minutes. The user input is the number of minutes.

Default format

```
<Param Name="RunTimeLimit" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="RunTimeLimit"
Caption="Max. run time (minutes):"/>
```

StackSizeLimit

Specifies the maximum stack segment size in kilobytes per process for each process that belongs to this job. The user input is the number of kilobytes.

Default format

```
<Param Name="StackSizeLimit" Type="FLOAT" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="StackSizeLimit"
Caption="Max. stack size (KB):"/>
```

Stdin

Specifies the fully-qualified path and name of the standard input file. The user input is a string.

Default format

```
<Param Name="Stdin" Type="FILE" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="Stdin" Caption="Standard in:" Type="FILE"/>
```

Stdout

Specifies the fully-qualified path and name of the standard output file. The user input is a string.

Default format

```
<Param Name="Stdout" Type="FILE" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="Stdout" Caption="Standard out:"/>
```

Stderr

Specifies the fully-qualified path and name of the standard error file. The user input is a string.

Default format

```
<Param Name="Stderr" Type="FILE" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="Stderr" Caption="Standard error:"/>
```

SubmissionCmd

Specifies the command that the job is to run. The user input is a string.

Default format

```
<Param Name="SubmissionCmd" Type="STRING" Expr="#{VALUE}"
 Required="TRUE" Repeatable="FALSE" Hidden="TRUE"/>
```

Example

```
<Param Name="SubmissionCmd" Caption="Command to run:"/>
```

UserGroup

Specifies the name of the fairshare group to associate this job with. The submitter must be a member of the specified group. The user input is a group name.

Default format

```
<Param Name="UserGroup" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="UserGroup" Caption="User group:"/>
```

UserName

Specifies the name of the user ID under which the job should run. The submitter must have administrative authority to set this value. The user input is a string.

Default format

```
<Param Name="UserName" Type="STRING" Expr="#{VALUE}"
Required="TRUE" Repeatable="FALSE" Hidden="FALSE"
Default="current_user"/>
```

Example

```
<Param Name="UserName" Caption="Run as user:"/>
```

UserPreExecCmd

Specifies a command to run on the execution host prior to running the job. This is typically used to set up the execution environment. The user input is a string.

Default format

```
<Param Name="UserPreExecCmd" Type="STRING" Expr="#{VALUE}"
 Required="TRUE" Repeatable="FALSE" Hidden="FALSE"/>
```

Example

```
<Param Name="UserPreExexCmd"
Caption="Preexecution command:"/>
```