

GASW Reference Manual

Tristan Glatard (glatard@creatis.insa-lyon.fr)

Silvia D. Olabbarriaga (S.D.Olabbarriaga@amc.uva.nl)

26 Feb 2009

Generic Application Service Wrapper (GASW)

<http://modalis.polytech.unice.fr/software/moteur/gasw>

The invocation of command line utilities is described in the following xml syntax:

```
<description>
<!-- describe executable file (one) -->
<executable name="name">
  <IN_FILE>
  [ <REQUIREMENT>]

<!-- describe input given as command line option (many) -->
  <input name="name" option="option_name">
    [ <IN_FILE> ]
  </input>

<!-- describe output given as command line option (many) -->
  <output name="name" option = "option_name">
    < OUT_FILE >
  </output>

<!-- other files that should be downloaded (many) -->
  <sandbox name="name">
    < IN_FILE >
  </sandbox>

</executable>
</description>
```

General notes:

- WN denotes the computer where the GASW service will invoke the execution of the described executable.
- “name” is an internal identifier (in general not used, except for **executable**)

Types of things that need to be described:

- **executable**: describes all aspects of an executable program. Need to indicate file with path. This file is downloaded on the current directory. The “name” is used to call the program with the command-line options!
- **input**: all inputs to the program (as command line parameters). Will generate options in the command line in the same order as they appear in the xml file. Need to indicate some string for **option_name**. A file is optional; if indicated, it will be downloaded in the

working directory and the basename given as command line option. If no file is indicated, a value given as workflow input will be used as command line option.

- **output**: all outputs of the program (as command line parameters). Need to indicate access type to file, file path template is optional. If template not informed, generates an LFN automatically. (*Check: also done for vfs?*)
- **sandbox**: optional, describes dependencies of the program, that is, all types of inputs (additional files with data and/or executables) that are not informed as command-line options. Need to indicate file with path. The file is downloaded in the working directory.

How to specify command-line options to the executable

Each input and output to the executable must be indicated via an option in the command line:

```
< input | output name="name" option = "option_name">  
    < INPUT | OUT_FILE >  
</ input | output>
```

`option_name = "-string | no<n> | > | 2> "`

Different types of options

- **-string**: will be used to generate a command line like this:
`<executable file> -string <value>`
Note that any string can be given, no error checking is performed.
All strings need to be different from each other. If <value> is a file, the basename is used.
- **no<n>**: no option is given, <n> needs to be different in each occurrence
- Special tricks to get output in std files:
 - Use ">" for stdout and define file name. Will generate:
`<executable file> > <file>`
 - Use "2>" for stderr, and define file name. Will generate:
`<executable file> 2> <file>`

Example (check):

```
<input name="input1" option="-opt1">  
</input>  
<input name="input2" option="no0">  
</input>
```

will generate a command line like this

```
<executable_name> -opt1 <value> <value2>
```

Describing files (<IN_FILE>, <OUT_FILE>)

A file is described by an "access type".

```
<access type= "URL" | "LFN" | "VFS" />  
[ < value | path | template=<file_path> ] />
```

Different **access types** (file access protocols) are possible:

- **URL:** HTTP and HTTPS protocol, no passwords. The file of given "name" is downloaded from the URL given by the path in the format <http://<hostname>:<port>/<dir>>
Only valid for input, files are downloaded with `wget`, all host certificates are trusted by default

Example: (check)

```
<... name = "myfile" >
<access type= "URL" >
    < path = "http://myserver.amc.nl:8080 /mydir/" />
</access>
<value value = "myfile.txt" />
```

- **LFN:** gLite logical file name, `file_path` specified with "value" in two ways:
 - gLite format: `/grid/<vo>/<path>`
Uses `LFC_HOST` and `LFC_HOME` variables as defined in MOTEUR.
 - URI format: `lfn://<host name>/grid/<vo>/<path>`
Here the given host name will be ignored, and the MOTEUR configuration will be used. (fix this, perhaps using `uri_copy.sh`?)
Files are downloaded/uploaded with gLite `lgc-cp` and `lcg-cr` commands.

Example:

```
<access type= "LFN" >
    < value = "/grid/vlmed/silvia/mydir/myfile.txt" />
</access>
```

- **VFS:** virtual file system (VL-e), various protocols (GridFTP, SRB, LFC). Given `file_path` corresponds to a VRL (Virtual Resource Locators) following the specification of URIs (Universal Resource Locators) as specified in RFC3986 (<http://www.ietf.org/rfc/rfc3986.txt>).
Downloaded with `uri_copy.sh`.

Syntax VRL/URI:

```
<SCHEME>: // [<USERINFO> @ ] <HOSTNAME> / <PATH>
```

```
<SCHEME>          ::= gftp | srb
<USERINFO>        ::= <USERNAME> . [<DOMAINNAME>]
<USERNAME>        ::= see RFC3986
<DOMAINNAME>     ::= see RFC3986
<HOSTNAME>        ::= see RFC3986
```

Notes:

- SRB URIs have a DOMAINNAME in the USERNAME part. VRLs will take the last dot separated string as domain name. For example in 'john.doe.vle', the 'vle' part is the domain name and 'john.doe' the username.
- LFNs have not been tested yet. Check/debug

- More types of VRL are valid (sftp), but however cannot be used with GASW - **check**

Example (value split in two lines to improve readability, should be single line):

```
<access type= "VFS" >
  < value = "srb://silvia.olabbarriaga.vlenl@srb.grid.sara.nl:50000/
  VLENL/home/silvia.olabbarriaga.vlenl/myfile.txt" />
</access>
```

Describing a file_path for inputs, executables and sandboxes (**IN_FILE**):

Need to specify a path

- **value**: complete path+file name is given, valid for LFN and VFS. For URL, contains name of the file.
- **path**: only for URL, http link is given
- Note: depending on the access type, configurations of the server and home directories are defined by MOTEUR (see `grid.conf` and `env.sh` files on MOTEUR installation)

Note: When running MOTEUR on "LOCAL" mode (in `grid.conf` file), all LFNs are interpreted as local files located on `$HOME/data/<value>`.

Describing a file_path for outputs (**OUT_FILE**):

Can optionally specify a template:

- **template**: Template to generate the output name.
- Special **variables** in template names
 - **%s**: automatically generates a number used to create unique identifiers for output files. Number is composed of 4 integers: time in seconds, time in ms, pid, random number. The integers are converted to digits and concatenated.
 - **\$na<n>**: use the name of the n-th input parameter described. If it is a file, use basename. Otherwise, use value of the parameter.
 - **\$dir<n>** use the directory of the file in the n-th input parameter described
 - n is the order of the input parameter in this xml (name), starts with zero.

When creating the output file, first try to delete the (possibly) existing file. After that registers the new file (`lcg-cr`). Fails component and workflow if cannot register file.

If template is not described, an automatic name will be generated `<number>.gen` (generated, on `LFC_HOME` - **check, better solution?**)

Describing dependencies for executable program

Two mechanisms:

- **Sandbox**: files that need to be downloaded, but that are not indicated as parameters in the command line
- **REQUIREMENT**: a grid-dependent parameter, specifies requirements for the job using gLite middleware GlueAttributes (see JDL attributes specification, <https://edms.cern.ch/file/590869/1/EGEE-JRA1-TEC-590869-JDL-Attributes-v0-9.pdf>)

Syntax: `<requirement value = "string" />`
will generate `Requirements = "string" ;`

Examples:

```
<requirement value = "other.GlueCEPolicyMaxWallClockTime > 1" \>
```

```
<requirement value = "Member(&quot;nl.v1-e.poc-release-2&quot;,  
other.GlueHostApplicationSoftwareRunTimeEnvironment) " />
```

Note: need to use XML syntax to encode special characters in string:
" = " (we can't use \" to put quotes in strings)

Check: [What happens with other requirements on grid.conf?](#)

Complete example:

```
Hello.pl -no0 <input file> -no1 <output file> 2> <stderr file>
```