Table of Contents

BLAH guide

BLAH Introduction

BLAHPD is a light component accepting commands according to the BLAH (Batch Local Ascii Helper) protocol to manage jobs on different Local Resources Management Systems (LRMS). The BLAH service provides a minimal, pragmatically designed common interface for job submission and control to a set of batch systems. Interaction with BLAH is achieved via text commands whose syntax and semantics is described below. Interaction with each one of the various supported batch system is achieved via customized scripts (the existing scripts are all Bourne shell scripts) that take care of executing the following functions:

- job submit - job hold - job resume - job status - job cancel

BLAH Portability

The purpose of this writeup is to:

- describe the assumptions that are currently made on the minimal functionality that batch systems must provide;
- describe the details of the command line arguments that are passed to each one of the five scripts that implement the fuctionality described above for a specific batch system;
- provide guidelines for porting of the existing scripts, especially the submit script for which helper shell functions were defined.

As the universe of different batch systems that need to be supported is very small, the most efficient way to get help beyond the information collected in these notes is to interact with the BLAH developers directly via the e-mail address blah-help@mi.infn.it

The string XXX, in the following text, is meant to be replaced with the name of the specific batch system being interfaced (pbs, lsf, condor, etc.).

Assumptions on batch system functionality.

BLAH assumes that batch systems are capable to

- Identify jobs for subsequent operations via a unique, constant identifier that is returned at the time of job submission.
- Transfer and optionally rename a configurable set of files from a given location on the submit node to the initial working directory on the selected worker node before the job execution starts.
- Transfer and optionally rename a configurable set of files from the initial working directory of jobs on the worker node that is selected for execution to a given location on the submit node.
- Provide status of currently running jobs via an appropriate command.
- Provide a historical track of jobs that were accepted and ran in the past via one or more log files.

BLAH can use this optional batch system functionality:

a) Hold (suspend) and resume jobs via appropriate commands.

BLAH doesn't require batch systems to

a) Be able to renew X509 proxies and/or transfer files to and from the worker node **while the job is being executed**. Proxy renewal is taken care of by a proxy renewal daemon that runs at the side of the running job and receives delegations of refreshed proxies during the lifetime of the job.

The supported batch systems

At the time of writing BLAH supports the following batch systems: LSF, PBS/Torque, SGE, Condor and just recently SLURM. In order to enable BLAH to interact with the selected batch system, it must be configured properly by setting few parameters on its configuration file which is located by default at /etc/blah.config. The file contains a list of parameters and their own default value is well defined for a standard setup of BLAH. Their customization is even possible if needed. For example to enable the support of SLURM is need just a specific parameter *slurm_binpath* which informs BLAH about the location where the SLURM executables are located (i.e. scontrol and sacct).

BLAH Scripts

XXX_submit.sh script

Submit a new job request to the batch system.

NOTE: most of the functionality for parsing and handling the submit script arguments is provided by a set of shell functions that are described further below (see SUBMIT SCRIPT HELPER FUNCTIONS). The argument description is provided for reference only.

SYNOPSIS:

XXX_submit.sh -c -q [-i] [-o] [-e] [-x] [-v | -V] [-s] [-d] [-w] [-n <# of MPI nodes>] [-r] [-p] [-l] [-j] [-T] [-I] [-O] [-R] [-C] [-- command_arguments]

Any argument after '--' will be passed to the user job ("command").

The command line switches have the following meaning (switches listed in alphabetical order):

[-C]: When this argument is present, the local script XXX_local_submit_attributes.sh is called, and its output is pasted to the batch system submit file. The XXX_local_submit_attributes.sh script is called after sourcing the contents of the 'CE requirements file' first. This file is composed by the BLAH main daemon and sets shell variables that specify attribute bounds derived from the CERequirements attribute in the BLAH submit command. Example format for the 'CE requirements file':

```
GlueHostMainMemoryRAMSize _Min=1000
GlueCEPolicyMaxCPUTime _Max=30
```

-c command: (shell) command to be executed as a batch job.

[-d]: Debug option. If set to 'yes', it causes the submit file to be sent to /dev/tty, and no actual submission to the batch system.

[-e]: Location of the local file that should receive the job STDERR upon job termination.

[-I]: points to an optional, temporary file containing a list of additional files (one per line) that need to be transferred from the submission node to the initial working directory on the worker node. This temp file is removed by the submit script.

[-i]: Location of a local file to be transferred to the worker node and connected to the STDIN of the job.

[-j]: String for unique identification of the job by the calling layer. As this string is unique, it is used to name the submit script, that will be created as 'cream_jobID'.

[-l]: Minimum remaining lifetime of the user proxy (expressed in seconds) before the proxy renewal damon will kill the user job. Defaults to 180 seconds, or 3 minutes.

[-n <# of MPI nodes>]: Number of MPI nodes to be reserved for the job, in case this feature is supported by the underlying batch system.

[-O]: points to an optional, temporary file containing a list of additional files (one per line) that need to be transferred from the working directory on the worker node back to the submit node. This temp file is removed by the submit script. The -R arguments can be used to establish file name remapping.

[-o]: Location of the local file that should receive the job STDOUT upon job termination.

[-p]: time interval (expressed in seconds) between attempts of the proxy renewal daemon to check that the use process is still alive. Defaults to 60 seconds.

-q Batch system queue to run the job in.

[-R]: Points to a file containing a list of names (one for each line of the file passed with the -O argument) that contain the local name of each each output file that is transferred from the worker node to the submit (local) node. The list file will be deleted by the submit script.

[-r]: Disables the entire BLAH proxy renewal machinery when set to 'no'. Defaults to 'yes' if missing.

[-T]: sets the directory location for storing temporary files.

[-v | -V]: Environment variables to be set for the user job. Two formats are possible: 1) semicolon-separated assignments -v "ENV1=val1;ENV2=val2;..." 2) space-separated assignments -V "ENV1=val1 ENV2=val2 ..."

[-x]: Location of the initial X509 user proxy to associate with teh job. This file is initially transferred with the job, then (optionally, see -r) renewed via proxy delegation.

[-w]: Directory location pre-pended to relative file paths and used as CWD by the submit script.

RETURN VALUE:

Termination code is zero on success, nonzero on error. Job identifier, with the identifier string 'BLAHP_JOBID_PREFIX' is returned on STDOUT on success. The job identifier must start with the batch system name followed by a slash (XXX/id).

SUBMIT SCRIPT HELPER FUNCTIONS:

A set of shell functions was written to ease the parsing and handling of the submit script options. They allow to write a submit script along the following template:

```
#!/bin/bash
# 1. Source definition of helper functions.
. `dirname $0`/blah_common_submit_functions.sh

# 2. Parse the submit options, that are used to set
#    a list of bls_opt_XYZ shell variables.
bls_parse_submit_options $@

# 3. Set up temporary files. Set a variable with the name of
#    an environment variable that holds the batch system job ID
#    at runtime.
bls_setup_all_files
bls_job_id_for_renewal=XXX_JOBID
```

```
# 4. Start writing the submit script to $bls_tmp_file

cat &gt; $bls_tmp_file &lt;&lt; end_of_preamble
#!/bin/bash
# PBS job wrapper generated by `basename $0`
# on `/bin/date`
# proxy_string = $bls_opt_proxy_string
# proxy_local_file = $bls_proxy_local_file
# Etc. Etc. Etc.
end_of_preamble

# 5. Handle script local customisations according to -C option
#    as appropriate for the batch system at hand.
if [ ! -z $bls_opt_req_file ] ; then
  echo \#\!/bin/sh &gt;&gt; ${bls_opt_req_file}-temp_req_script
  cat $bls_opt_req_file &gt;&gt; ${bls_opt_req_file}-temp_req_script
  echo "source ${GLITE_LOCATION:-/opt/glite}/bin/XXX_local_submit_attributes.sh" &gt;&gt; ${b
  chmod +x ${bls_opt_req_file}-temp_req_script
  ${bls_opt_req_file}-temp_req_script  &gt;&gt; $bls_tmp_file 2&gt; /dev/null
  rm -f ${bls_opt_req_file}-temp_req_script
  rm -f $bls_opt_req_file
fi

# 6. Add specific directives to select queue ($bls_opt_queue) and
#    MPI node request ($bls_opt_mpinodes)

# 7. Add directives to transfer and rename input and output files.
#    These are stored as
#    $bls_inputsand_local_0...$bls_inputsand_local_n-1
#    $bls_inputsand_remote_0...$bls_inputsand_remote_n-1
#    $bls_outputsand_local_0...$bls_outputsand_local_n-1
#    $bls_outputsand_remote_0...$bls_outputsand_remote_n-1
#
#    Two shell functions can help here.
#    a:
#      bls_fl_subst_and_accumulate inputsand "@@F_REMOTE/@@F_LOCAL" "sep"
#      bls_fl_subst_and_accumulate outputsand "@@F_REMOTE/@@F_LOCAL" "sep"
#      fill $bls_fl_subst_and_accumulate_result with a list of "sep"
#      separated strings formatted as shown in the second argument.
#      The submit node full file path is substituted to @@F_LOCAL
#      and the worker node path relative to the initial working dir
#      is substituted to @@F_REMOTE.
#    b:
#      bls_fl_subst_and_dump inputsand "@@F_LOCAL&gt;@@F_REMOTE" $bls_tmp_file
#      bls_fl_subst_and_dump outputsand "@@F_LOCAL&lt;@@F_REMOTE" $bls_tmp_file
#      append to $bls_tmp_file a line for each input and output file,
#      where @@F_REMOTE and @@F_LOCAL are substituted as above.

# 8. Append job wrapper as a shell script to $bls_tmp_file
bls_add_job_wrapper

# 9. Send the submit file $bls_tmp_file to the batch system and
#    try making sure it doesn't get lost.

# 10. Echo to STDOUT the unique job ID to be used by subsequent scripts
#     (with BLAHP_JOBID_PREFIX) and wrap up. The job ID must be
#     properly understood by subsequent commands.
echo "BLAHP_JOBID_PREFIXXXX?$jobID"
bls_wrap_up_submit
exit $retcode
```

XXX_hold.sh script

Suspend the execution of a job.

SYNOPSIS:

XXX_hold.sh The job identifier must be the same string returned by the submit script (without the leading BLAHP_JOBID_PREFIX). Any leading part up to the first slash '/' will be ignored by the script.

RETURN VALUE:

Termination code is zero on success, nonzero on error.

XXX_resume.sh script

Resume the execution of a (previously suspended) job.

SYNOPSIS:

XXX_resume.sh The job identifier must be the same string returned by the submit script (without the leading BLAHP_JOBID_PREFIX). Any leading part up to the first slash '/' will be ignored by the script.

RETURN VALUE:

Termination code is zero on success, nonzero on error.

XXX_status.sh script

Get current status of a job.

SYNOPSIS:

XXX_status.sh [-w] [-n] The job identifier must be the same string returned by the submit script (without the leading BLAHP_JOBID_PREFIX). Any leading part up to the first slash '/' will be ignored by the script.

[-n] Option specific to the CREAM caller. Return the port used by the BLParser (see description below) to communicate with the CREAM service

[-w] Return the worker node where a job is running as an attribute of the output classad (WorkerNode ="host.domain")

RETURN VALUE:

The script must return a string-formatted classad (see http://www.cs.wisc.edu/condor/classad/refman/ for the complete reference on classad syntax) containing at least the following attributes:

- BatchjobId = "jobId (without the leading batch system name)"
- JobStatus = status_code (1 = IDLE, 2 = RUNNING, 3 = REMOVED, 4 = COMPLETED 5 = HELD) (only for COMPLETED jobs): ExitCode = code

Example of status script results:

(job queued) [BatchjobId = "26526.atlfarm006.mi.infn.it"; JobStatus = 1]

(job completed) [ExitCode = 0; BatchjobId = "26526.atlfarm006.mi.infn.it"; JobStatus = 4]

XXX_cancel.sh script

Remove a job from the batch system queue.

SYNOPSIS:

XXX_cancel.sh The job identifier must be the same string returned by the submit script (without the leading BLAHP_JOBID_PREFIX). Any leading part up to the first slash '/' will be ignored by the script.

RETURN VALUE:

Termination code is zero on success, nonzero on error.

BLAH forward requirements to the local batch system

The user can set some requirements to be forwarded to the local batch system, by using the attribute 'CERequirements', as defined above, in the 'blah_job_submit' command. This can be achieved both with direct submission to the CREAM CE and with submission to the CE via the WMS, as explained in the following:

- direct submission to CREAM -> the attributes to be forwarded are specified in the .jdl 'CERequirements' attribute and are the ones of the GlueSchema in use
- submission to a CE via WMS -> the CERequirements attribute for 'blah_job_submit' is filled taking into account the value of the job JDL Requirements expression and what is specified as CeForwardParameters in the WMS conf file (workloadmanager section). Also in this case the parameters to be forwarded are chosen from the GlueSchema in use

BLAH Commands syntax and semantics

BLAH Commands

The following list of commands represents the set of commands required for interaction with the BLAHP server, interfacing to a given Local Resource Management system. This is based on the minimum set of commands used in the original GAHP (v1.0.0) specification removing commands that are specific to the operation of the GRAM protocol (INITIALIZE_FROM_FILE, GASS_SERVER_INIT, GRAM_CALLBACK_ALLOW, GRAM_JOB_CALLBACK_REGISTER, GRAM_PING). The JOB_SIGNAL command may be initially left unimplemented for some of the batch systems (and in that case will return an error -E- state and will not be returned by COMMANDS).

- BLAH_JOB_CANCEL
- BLAH_JOB_SIGNAL
- BLAH_JOB_HOLD
- BLAH_JOB_REFRESH_PROXY
- BLAH_JOB_RESUME
- BLAH_JOB_STATUS
- BLAH_JOB_STATUS_ALL
- BLAH_JOB_STATUS_SELECT
- BLAH_JOB_SUBMIT
- BLAH_SET_GLEXEC_DN
- BLAH_SET_GLEXEC_OFF
- COMMANDS
- CACHE_PROXY_FROM_FILE
- QUIT
- RESULTS

- USE_CACHED_PROXY
- UNCACHE_PROXY
- VERSION

Optionally, the following two commands may also be implemented:

- ASYNC_MODE_ON
- ASYNC_MODE_OFF

BLAHP Commands structure

Conventions and Terms used

Below are definitions for the terms used in the sections to follow:

<CRLF>The characters carriage return and line feed (in that order), *or* solely the line feed character.

<SP>The space character.

line A sequence of ASCII characters ending with a <SP>

Request Line A request for action on the part of the BLAHP server.

Return Line A line immediately returned by the BLAHP server upon receiving a Request Line.

Result Line A line sent by the BLAHP server in response to a RESULTS request, which communicates the results of a previous asynchronous command Request.

S: and **R:** In the Example sections for the commands below, the prefix "S: " is used to signify what the client sends to the BLAHP server. The prefix "R: " is used to signify what the client receives from the BLAHP server. Note that the "S: " or "R: " should not actually be sent or received.

Commands structure

BLAHP commands consist of three parts:

- Request Line
- Return Line
- Result Line

Each of these "Lines" consists of a variable length character string ending with the character sequence <CRLF>.

A Request Line is a request from the client for action on the part of the BLAHP server. Each Request Line consists of a command code followed by argument field(s). Command codes are a string of alphabetic characters. Upper and lower case alphabetic characters are to be treated identically with respect to command codes. Thus, any of the following may represent the blah_job_submit command: blah_job_submit Blah_Job_Submit blAh_joB_suBMit BLAH_JOB_SUBMIT In contrast, the argument fields of a Request Line are _case sensitive_.

The Return Line is always generated by the server as an immediate response to a Request Line. The first character of a Return Line will contain one the following characters: S - for Success F - for Failure E - for a syntax or parse Error Any Request Line which contains an unrecognized or unsupported command, or a command with an insufficient number of arguments, will generate an "E" response.

The Result Line is used to support commands that would otherwise block. Any BLAHP command which may require the implementation to block on network communication require a "request id" as part of the Request Line. For such commands, the Result Line just communicates if the request has been successfully parsed and queued for service by the BLAHP server. At this point, the BLAHP server would typically dispatch a new thread to actually service the request. Once the request has completed, the dispatched thread should create a Result Line and enqueue it until the client issues a RESULT command.

Transparency

Arguments on a particular Line (be it Request, Return, or Result) are typically separated by a <SP>. In the event that a string argument needs to contain a <SP> within the string itself, it may be escaped by placing a backslash ("\") in front of the <SP> character. Thus, the character sequence "\ " (no quotes) must not be treated as a separator between arguments, but instead as a space character within a string argument.

Sequence of Events

Upon startup, the BLAHP server should output to stdout a banner string which is identical to the output from the VERSION command without the beginning "S " sequence (see example below). Next, the BLAHP server should wait for a complete Request Line from the client (e.g. stdin). The server is to take no action until a Request Line sequence is received.

Example:

```
R: $GahpVersion: x.y.z Feb 31 2004 INFN\ Blahpd $
 S: COMMANDS
 R: S COMMANDS BLAH_JOB_CANCEL BLAH_JOB_SIGNAL BLAH_JOB_STATUS BLAH_JOB_SUBMIT COMMANDS QUIT RES
 S: VERSION
 R: S $GahpVersion: x.y.z Feb 31 2004 INFN\ Blahpd $
        (other commands)
 S: QUIT
 R: S
```

BLAH Commands syntax

This section contains the syntax for the Request, Return, and Result line for each of the following commands:

- **COMMANDS**: List all the commands from this protocol specification which are implemented by this BLAHP server.
  + Request Line: COMMANDS <CRLF>
  + Return Line: S <SP> <SP> <SP> ... <CRLF>
  + Result Line: None.

- **VERSION**: Return the version string for this BLAHP. The version string follows a specified format (see below). Ideally, the version entire version string, including the starting and ending dollar sign ($) delimiters, should be a literal string in the text of the BLAHP server executable. This way, the Unix/RCS "ident" command can produce the version string. The version returned should correspond to the version of the protocol supported.
  + Request Line: VERSION <CRLF>
  + Return Line:
    - S <SP> $GahpVesion: <SP> .. <SP> <build-month _moz-userdefined=""> <SP> <build-day-of-month _moz-userdefined=""> <SP> <build-year _moz-userdefined=""> <SP> <general-descrip _moz-userdefined=""> <SP>$ <CRLF> </general-descrip></build-year></build-day-of-month></build-month>
    - major.minor.subminor = for this version of the protocol, use version 1.0.0.
    - build-month = string with the month abbreviation when this BLAHP server was built or released. Permitted values are: "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",

"Sep", "Oct", "Nov", and "Dec".
    - build-day-of-month = day of the month when BLAHP server was built or released; an integer between 1 and 31 inclusive.
    - build-year = four digit integer specifying the year in which the BLAHP server was built or released.
    - general-descrip = a string identifying a particular BLAHP server implementation.

+ Result Line: None.
+ Example:

```
S: VERSION R: S $GahpVersion: x.y.z Feb 31 2004 INFN\ Blahpd $
```

- **QUIT**: Free any/all system resources (close all sockets, etc) and terminate as quickly as possible.
  + Request Line: QUIT <CRLF>
  + Return Line: S <CRLF>
  Immediately afterwards, the command pipe should be closed and the BLAHP server should terminate.
  + Result Line: None.

- **RESULTS**: Display all of the Result Lines which have been queued since the last RESULTS command was issued. Upon success, the first return line specifies the number of subsequent Result Lines which will be displayed. Then each result line appears (one per line) -- each starts with the request ID which corresponds to the request ID supplied when the corresponding command was submitted. The exact format of the Result Line varies based upon which corresponding Request command was issued.
  IMPORTANT: Result Lines must be displayed in the *exact order* in which they were queued!!! In other words, the Result Lines displayed must be sorted in the order by which they were placed into the BLAHP's result line queue, from earliest to most recent.
  + Request Line: RESULTS
  + Return Line(s): S <SP><num-of-subsequent-result-lines _moz-userdefined=""> <CRLF> <SP> ... <CRLF> <SP> ... <CRLF>...
  * reqid = integer Request ID, set to the value specified in the corresponding Request Line.
  + Result Line: None.
  + Example:

```
S: RESULTS
  R: S 1
  R: 100 0
```

-
    - **ASYNC_MODE_ON**: Enable Asynchronous notification when the BLAHP server has results pending for a client. This is most useful for clients that do not want to periodically poll the BLAHP server with a RESULTS command. When asynchronous notification mode is active, the GAHP server will print out an 'R' (without the quotes) on column one when the 'RESULTS' command would return one or more lines. The 'R' is printed only once between successive 'RESULTS' commands. The 'R' is also guaranteed to only appear in between atomic return lines; the 'R' will not interrupt another command's output.
      If there are already pending results when the asynchronous results available mode is activated, no indication of the presence of those results will be given. A GAHP server is permitted to only consider changes to it's result queue for additions after the ASYNC_MODE_ON command has successfully completed. GAHP clients should issue a 'RESULTS' command immediately after enabling asynchronous notification, to ensure that any results that may have been added to the queue during the processing of the ASYNC_MODE_ON command are accounted for.
      + Request Line: ASYNC_MODE_ON <CRLF>

+ Return Line: S <CRLF> Immediately afterwards, the client should be prepared to handle an R <CRLF>appearing in the output of the GAHP server.
+ Result Line: None.
+ Example:

```
  S: ASYNC_MODE_ON
  R: S
  S: BLAH_JOB_CANCEL 00001 123.bbq.mi.infn.it
  R: S
  S: BLAH_JOB_CANCEL 00002 124.bbq.mi.infn.it
  R: S
  R: R
  S: RESULTS
  R: S 2
  R: 00001 0
  R: 00002 0  R: 00002 0  R: 00002 0  R: 00002 0  R: 00002 0  R: 00002 0  R: 00002 0  R: 00002 0
```

Note that you are NOT guaranteed that the 'R' will not appear between the dispatching of a command and the return line(s) of that command; the GAHP server only guarantees that the 'R' will not interrupt an in-progress return. The following is also a legal example:

```
S: ASYNC_MODE_ON
  R: S
  S: BLAH_JOB_CANCEL 00001 123.bbq.mi.infn.it
  R: S
  S: BLAH_JOB_CANCEL 00002 124.bbq.mi.infn.it
  R: R
  R: S
  S: RESULTS
  R: S 2
  R: 00001 0
  R: 00002 0
```

-
  - **ASYNC_MODE_OFF**: Disable asynchronous results-available notification. In this mode, the only way to discover available results is to poll with the RESULTS command. This mode is the default. Asynchronous mode can be enabled with the ASYNC_MODE_ON command.
    + Request Line: ASYNC_MODE_OFF <CRLF>

+ Return Line: S <CRLF> + Results Line: None
+ Example:

```
  S: ASYNC_MODE_OFF
  R: S
```

BLAH_JOB_SUBMIT

Submit a job request to a specified queue (specified in the submit classad). This will cause the job to be submitted to the batch system.
+ Request Line: BLAH_JOB_SUBMIT <SP> <reqid> <SP> <submit classad> <CRLF>

-
  - reqid = non-zero integer Request ID
  - submit classad = valid submit description for the job, in string representation. See paragraph 3.0 for a description of the format. Here's a list of supported attributes with a brief description.
    "Cmd"
        Full path of the executable in the local filesystem
    "Args"
        List of individual arguments (no '/bin/sh' convention on argument separation, but separate arguments) for the executable

"In"
: Full path in the local filesystem where the standard input for the executable is found

"Out"
: Full path in the local filesystem where the standard output of the executable will be stored (at job completion).

"Err"
: Full path in the local filesystem where the standard error of the executable will be stored (at job completion).

"X509UserProxy"
: Full path wherethe proxy certificate is stored.

"Env"
: Semicolon-separated list of environment variables of the form:

```
<parameter> = <value>
```

"Stagecmd" :Sets if the executable of the job must be copied on the WorkerNode
: can be "TRUE" or "FALSE".

"Queue"
: Queue in the local batch system where the job must be enqueued.

"Gridtype"
: String indicating the underlying local batch system (currently "pbs" and "lsf" supported).

"uniquejobid"
: unique name identifier for the final job to be submitted to the local batch system.

"NodeNumber"
: number of nodes to be reserved for an MPI job. It gets translated into the LRMS command qsub "-l nodes=" for pbs or into "-n " for lsf. For Condor this attribute is not supported, therefore it gets ignored.

"CERequirements"
: string containing the requirements to be forwarded to the local batch system

"TransferInput = file1,file2,file..."
: comma-delimited list of all the files to be transferred into the working directory for the job before the job is started. Only the transfer of files is available. The transfer of subdirectories is not supported.

"TransferOutput = file1,file2,file..."
: an explicit list of output files to be transferred back from the temporary working directory on the execute machine to the submit machine (where BLAHPD is running). Only the standard output and standard error files, are transferred back by default if requested.

"TransferOutputRemaps = name1=newname1;name2=newname2;name=newname..."
: This attribute specifies the name (and optionally the complete path) to use when downloading output files from the completed job. **Note that the mappings are separated by semicolons**. Normally, output files are transferred back to the initial working directory with the same name they had in the execution directory. This gives you the option to save them with a different path or name. If you specify a relative path, the final path will be relative to the job's initial working directory (the directory specified in the 'Iwd' attribute, or, if this is missing, the current working directory of BLAHPD)

+ Return Line:

```
<result> <CRLF>
```

-
    - result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments

(e.g. an unrecognized or unsupported command, or for missing or malformed arguments).
+ Result Lines: <reqid> <sp> <result-code> <sp> <error-string> <sp> <job_local_id> <crlf>

  - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
  - result-code = integer equal to 0 on success, or an error code
  - error-string = description of error
  - job_local_id = on success, a string representing a unique identifier for the job. This identifier must not be bound to this BLAHP server, but instead must be allowed to be used in subsequent BLAHP server instantiations. For instance, the job_local_id must be implemented in such a fashion that the following sequence of events by the caller must be permissible:
    - a) issue a BLAH_JOB_SUBMIT command
    - b) read the job_local_id in the result line
    - c) store the job_local_id persistently
    - d) subsequently kill and restart the BLAHP server process * e) issue a BLAH_JOB_CANCEL command, passing it the stored job_local_id value obtained in step (b).
      + Example:

```
       S: BLAH_JOB_SUBMIT 2 [\ Cmd\ =\ "/usr/bin/test.sh";\ Args\ =\ "'X=3:Y=2'";
   \ Env\ =\ "VAR1=56568";\ In\ =\ "/dev/null";\ Out\ =\ "/home/StdOutput";
   \ Err\ =\ "/home/error";\ x509userproxy\ =\ "/home/123.proxy";\ Stagecmd
         \ =\ TRUE;\ Queue\ =\ "short";\ GridType\ =\ "pbs";\ ]'
R: S
S: RESULTS
R: 2 0 No\ error pbs/20051012/2957
```

BLAH_JOB_CANCEL

This function removes an IDLE job request, or kill all processes associated with a RUNNING job, releasing any associated resources.
+ Request Line: BLAH_JOB_CANCEL <sp> <reqid> <sp> <job_local_id> <CRLF>

-
  - reqid = non-zero integer Request ID
  - job_local_id = job_local_id (as returned from BLAH_JOB_SUBMIT) of the job to be canceled.

+ Return Line: <result> <crlf>

-
  - result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments (e.g. an unrecognized or unsupported command, or for missing or malformed arguments).

+ Result Line: <reqid> <sp> <result-code> <sp> <error-string> <CRLF>

-
  - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
  - result-code = integer equal to 0 on success, or an error code
  - error-string = description of error
    + Example:

```
S: BLAH_JOB_CANCEL 1 pbs/20051012/2957.grid001.mi.infn.it
R: S
R: R
S: RESULTS
R: S 1
R: 1 0 No\ error
```

BLAH_JOB_STATUS

Query and report the current status of a submitted job.
+ Request Line: BLAH_JOB_STATUS <sp> <reqid> <sp> <job_local_id> <CRLF>

-
    - reqid = non-zero integer Request ID
    - job_local_id = job_local_id (as returned from BLAH_JOB_SUBMIT) of the job whose status is desired.

+ Return Line: <result> <CRLF>

-
    - result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments (e.g. an unrecognized or unsupported command, or for missing or malformed arguments).

+ Result Line: <reqid> <sp> <result-code> <sp> <error-string> <sp> <job_status> <sp> <result-classad> <CRLF>

-
    - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
    - result-code = integer equal to 0 on success, or an error code
    - error-string = description of error
    - job_status = if the result_code is 0 (success), then job_status is set to an integer based upon the status of the job as follows:
        - 1 IDLE (job is waiting on the batch system queue)
        - 2 RUNNING (job is executing on a worker node)
        - 3 REMOVED (job was successfully cancelled)
        - 4 COMPLETED (job completed its execution on the batch system)
        - 5 HELD (job execution is suspended; job is still in the batch system queue)
    - result-classad = Aggregate information about the job status. The classad format can vary with the local batch system. Typically, the following attributes are defined:

JobStatus	job status - same codes as described above
BatchjobId	Job ID as known to the local batch system
ExitCode	Termination code - only for finished jobs
ExitReason	Exit reason, if available - only for finished jobs
WorkerNode	When available, FQDN of the worker node - only for running jobs

Example:

```
S: BLAH_JOB_STATUS 1 pbs/20051012/2958.grid001.mi.infn.it
R: S
R: R
S: RESULTS
R: S 1
R: 1 0 No\ Error 2 [\ BatchjobId\ =\ "2958.grid001.mi.infn.it";
   \ JobStatus\ =\ 2;\ WorkerNode\ =\ "\ grid001.mi.infn.it"\ ]
```

BLAH_JOB_STATUS_ALL

This command is only available if the BLAH local job registry file is configured in the BLAH config file (job_registry attribute) and supported by the active batch system. Query and report the current status of all jobs managed by the BLAH server.

+ Request Line: BLAH_JOB_STATUS_ALL <sp> <reqid> <CRLF>

-
    - reqid = non-zero integer Request ID

+ Return Line: <result> <CRLF>

-
    - result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments (e.g. an unrecognized or unsupported command, or for missing or malformed arguments).

+ Result Line: <reqid> <sp> <result-code> <sp> <error-string> <sp> <result-classad> <CRLF>

-
    - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
    - result-code = integer equal to 0 on success, or an error code
    - error-string = description of error
    - result-classad = List of classads containing aggregate information about the job status. In addition to the attributes defined for the output of the BLAH_JOB_STATUS command, as the status info comes from a local cache, the following attributes may also be present:
        - BlahJobId: Job ID as known to the BLAH layer. Typically a decorated form of BatchJobId
        - CreateTime: Seconds since the Unix epoch at the time the first job info was inserted into the cache.
        - ModifiedTime: Seconds since the Unix epoch when the most recent modification of the job info occurred.

BLAH_JOB_STATUS_SELECT

This command is only available if the BLAH local job registry file is configured in the BLAH config file (job_registry attribute) and supported by the active batch system. Query and report the current status of all jobs managed by the BLAH server.
+ Request Line: BLAH_JOB_STATUS_SELECT <sp> <reqid> <sp> <selection expression> <CRLF>

-
    - reqid = non-zero integer Request ID
    - selection expression = Classad expression defining select requirements on the returned job ads (e.g.: JobStatus ==2).

+ Return Line: <result> <crlf>

-
    - result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments (e.g. an unrecognized or unsupported command, or for missing or malformed arguments).

+ Result Line: <reqid> <sp> <result-code> <sp> <error-string> <sp> <result-classad> <CRLF>

-
    - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
    - result-code = integer equal to 0 on success, or an error code
    - error-string = description of error
    - result-classad = List of classads containing aggregate information about the status of the jobs that satisfy the selection requirement. The format is the same as for the BLAH_JOB_STATUS_ALL command.

BLAH_JOB_SIGNAL

Send a signal (if possible) to a specified job. This has to be in the RUNNING status.
+ Request Line: BLAH_JOB_SIGNAL <sp> <reqid> <sp> <job_local_id> <sp> <signal> <CRLF>

-
    - reqid = non-zero integer Request ID
    - job_local_id = job_local_id (as returned from BLAH_JOB_SUBMIT) of the job whose status is desired.
    - signal = an integer with the signal to send

+ Return Line:<result> <CRLF>

-
    - result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments (e.g. an unrecognized or unsupported command, or for missing or malformed arguments).

+ Result Line: <reqid> <sp> <result-code> <sp> <error-string> <sp> <job_status> <CRLF>

-
    - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
    - result-code = integer equal to 0 on success, or an error code
    - error-string = description of error
    - job_status = if the result_code is 0 (success), then job_status is set to an integer based upon the status of the job as follows (compare above):
        - ◊ 1.IDLE
        - ◊ 2. RUNNING
        - ◊ 3. REMOVED
        - ◊ 4. COMPLETED
        - ◊ 5. HELD

BLAH_JOB_REFRESH_PROXY

Renew the proxy of an already submitted job. The job has to be in IDLE, RUNNING or HELD status.
+ Request Line: BLAH_JOB_REFRESH_PROXY <sp> <reqid> <sp> <job_local_id> <sp> <proxy_file> <CRLF>

-
    - reqid = non-zero integer Request ID
    - job_local_id = job_local_id (as returned from BLAH_JOB_SUBMIT) of the job whose proxy has to be renewed.
    - proxy_file = path to the fresh proxy file.

+ Return Line: <result> <CRLF>

-
    - result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments (e.g. an unrecognized or unsupported command, or for missing or malformed arguments).
      + Result Line: <reqid> <sp> <result-code> <sp> <error-string> <crlf>
    - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
    - result-code = integer equal to 0 on success, or an error code
    - error-string = description of error
      Example:

```
S: BLAH_JOB_REFRESH 1 123.proxy
R: S
R: R
S: RESULTS
R: S 1
```

```
R: 1 0 No\ Error
```

BLAH_JOB_HOLD

This function always puts an IDLE job request in a HELD status. If the job is already running RUNNING it can be HELD too, depending whether the underlying batch system supports this feature.
+ Request Line: BLAH_JOB_HOLD <sp> <reqid> <sp> <job_local_id> <crlf>

-
    - reqid = non-zero integer Request ID
    - job_local_id = job_local_id (as returned from BLAH_JOB_SUBMIT) of the job to be canceled.

+ Return Line: <result> <crlf> result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments (e.g. an unrecognized or unsupported command, or for missing or malformed arguments).
+ Result Line: <reqid> <sp> <result-code> <sp> <error-string> <crlf>

-
    - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
    - result-code = integer equal to 0 on success, or an error code
    - error-string = description of error
      + Example:

```
S: BLAH_JOB_HOLD 1 pbs/20051012/2957.grid001.mi.infn.it
R: S
R: R
S: RESULTS
R: S 1
R: 1 0 No\ error
```

BLAH_JOB_RESUME

This function puts an HELD job request in the status it was before the holding action.
+ Request Line: BLAH_JOB_RESUME <sp> <reqid> <sp> <job_local_id> <crlf>

-
    - reqid = non-zero integer Request ID
    - job_local_id = job_local_id (as returned from BLAH_JOB_SUBMIT) of the job to be canceled.

+ Return Line: <result> <crlf>

-
    - result = the character "S" (no quotes) for successful submission of the request (meaning that the request is now pending), or an "E" for error on the parse of the request or its arguments (e.g. an unrecognized or unsupported command, or for missing or malformed arguments).

+ Result Line: <reqid> <sp> <result-code> <sp> <error-string> <crlf>

-
    - reqid = integer Request ID, set to the value specified in the corresponding Request Line.
    - result-code = integer equal to 0 on success, or an error code
    - error-string = description of error
      + Example:

```
S: BLAH_JOB_RESUME 1 pbs/20051012/2957.grid001.mi.infn.it
R: S
R: R
S: RESULTS
R: S 1
R: 1 0 No\ error
```

BLAH_SET_GLEXEC_DN

This function enables local user mapping via GLEXEC for all subsequent actions. The call is synchronous and the effect is immediate.
+ Request Line: BLAH_SET_GLEXEC_DN <sp> <user_proxy> <sp> <ssl_client_certificate> <sp> <not_limited> <crlf>

-
  - user_proxy Path to a valid GSI proxy file. This will be optionally (when not_limited == 0) turned into a limited (delegated) proxy and be made available via GLEXEC to the user job. (The delegated proxy path is set in the GLEXEC_SOURCE_PROXY enviroment variable prior to the GLEXEC call).
  - ssl_client_certificate Path to a valid GSI proxy file. Can be the same as user_proxy. This is the client certificate that is used for access to GLEXEC and local user mapping (it is set in the GLEXEC_CLIENT_CERT environment variable prior to the GLEXEC call).
  - not_limited If this argument is set to a numeric value not equal to zero, the supplied proxy will not be limited (delegated).

+ Return Line:
One of the following: S <sp> <action description> <crlf> - in case of success F <crlf> - in case of failure
+ Result Line: None.
+ Example:

```
  S: BLAH_SET_GLEXEC_DN /path/test.proxy /path/test.proxy 0
  R: S Glexec\ mode\ on
```

BLAH_SET_GLEXEC_OFF

This command unconditionally disables user mapping via GLEXEC for all subsequent actions. The call is synchronous and the effect is immediate.
+ Request Line: BLAH_SET_GLEXEC_OFF <crlf>

-
  - reqid = integer Request ID, set to the value specified in the corresponding Request Line.

+ Return Line: One of the following: S <sp> <action-description> <crlf> - in case of success F <crlf> - in case of failure
+ Result Line: None.
+ Example:

```
S: BLAH_SET_GLEXEC_OFF
  R: S Glexec\ mode\ off
```

CACHE_PROXY_FROM_FILE

This command is only available if enable_glexec_from_condor is set to 'yes' in the BLAHPD config file. Read a GSI proxy from a specified file, and store it under the specified symbolic name.
Unlike other GAHPs, and because this function is only used to trigger GLEXEC access which requires access to a proxy file, BLAH does not read and cache the proxy contents but only the file name. Changes to the file will affect the use of the stored proxy. The proxy name caching is lost when the BLAHP server terminates.
The intent of this command is to allow the BLAHP server to capture and use different proxy files for different jobs.
Other commands (e.g. by USE_CACHED_PROXY) may later be reference the given cached proxy file by its symbolic name. USE_CACHED_PROXY has the side effect of enabling GLEXEC.
+ Request Line:CACHE_PROXY_FROM_FILE <sp> <id> <sp> <path-to-local-proxy-file> <crlf>

-
  - <id> = a symbolic name which will be used to reference the given proxy.

- ♦ <path-to-local-proxy-file> = a fully-qualified pathname to a file local to the BLAHP server which contains a valid GSI proxied certificate.
  + Return Line: One of the following:
  S <sp> <action-description> <crlf&gtl;
  F <sp> <error-string> <crlf>Upon success, use the "S" version; if not recognized, use the "F" version.
- ♦ error-string = brief string description of the error, appropriate for reporting to a human end-user.
  + Result Line:None.
  + Example:

```
S: CACHE_PROXY_FROM_FILE 1 /path/test.proxy
R: S Proxy\ cached
```

USE_CACHED_PROXY

This command is only available if enable_glexec_from_condor is set to 'yes' in the BLAHPD config file. Sets the proxy previously cached under the specified name, as the "active" proxy. Enable user mapping via GLEXEC for all subsequent actions. The active proxy will be used for GLEXEC user mapping, it will remain active until it is changed via the next invocation of USE_CACHED_PROXY.
The proxy must have been previously cached (under the specified name) using CACHE_PROXY_FROM_FILE command.
This command allows the BLAHP server to simultaneously act on behalf of two or more jobs that require
+ Request Line: USE_CACHED_PROXY <sp> <id> <crlf> <id> = the symbolic name of the proxy to use
+ Return Line: One of the following:
S <sp> <action-description> <crlf> F <sp> <error-string> <crlf> Upon success, use the "S" version; if not recognized, use the "F" version.

-
  - ♦ error-string = brief string description of the error, appropriate for reporting to a human end-user.
    + Result Line: None.
    + Example:

```
S: USE_CACHED_PROXY 1
R: S Glexec\ proxy\ set\ to\ /path/test.proxy\ (dir\ IWGRP\ bit\ already\ on)
```

BLParser

To obtain efficient access and prompt update to the current status of BLAH active jobs, a 'log parser' daemon was developed for LSF and PBS. The submit and status scripts can make optional use of the batch log parser to increase their efficiency.

Here's a description of the semantics of the batch log parser (BLParser).

The BLParser listens on a network socket and replies to a set of mandatory and a set of optional queries (optional queries can be used for info and debug).

Here is the list of mandatory queries and their replies:

- BLAHJOB/<blahjob-id> <lrms-jobid>

- <date-yyyymmdd>/<lrms-jobid> a classad like this: [BatchJobId=<lrms-jobid> Workernode=<workernode> JobStatus =<jobstatus> LRMSSubmissionTime =<submission time> LRMSStartRunningTime =<start time> LRMSCompletedTime =<finish time=> ExitReason =<exitreason> ExitCode =<exit code=>]/pr_removal where pr_removal is a flag that told the status script if the proxy file has to be removed (when job is killed or finished) or not.

- CREAMPORT => port where cream can connect to the parser.

- TEST => print Y<lrmsname> (e.g. YLSF YPBS). This query is used to know if a parser is active. This is useful when in config file is specified more than one parser to try: if the first does not reply to the TEST query the next one is tried.

These are optional queries and their replies:

- HELP => print command list

- VERSION => print version

- TOTAL => print total number of jobs cached in the parser

- LISTALL => print jobid of all jobs cached in the parser

- LISTF[/<first-n-jobid>] => print first n jobid </first-n-jobid>

- LISTL[/<last-n-jobid>] => print last n jobid

The BLParser includes an option to notify to a client (typically the CREAM service) every state change in CREAM-managed jobs. CREAM sends to the parser a string like this:

STARTNOTIFY/<date-yyyymmdd>

atfer that string the parser sends back to cream every state change for cream job since the date contained in the string with lines like this:

NTFDATE/<classad>

where <classad> is a classad like the one sent in reply to <date-yyyymmdd>/<lrms-jobid> query.

After that parser will sent to cream every new state change for cream jobs.

Cream can change the default prefix (cream_) used to identify tracked jobs among all other jobs with this command:

CREAMFILTER/<creamfilter>

where <creamfilter> should be a string like crXXX_ (XXX can be any character [a-z],[A-Z],[0-9]). This command has to sent before STARTNOTIFY.

There is a startup script for the BLParser that reads all the parameters from a config file. It is possible to set the debug level, the debug log file, the log file location; it is also possible to start different parsers listening on different ports with the same startup script.

New Parser Scenario

The old BLParser is now splitted in two different daemons:

- BUpdater (that is specific for each batch system:there are BUpdaterLSF, BUpdaterPBS and BUpdaterCondor)
- BNotifier that is the same for all batch systems.

Both daemons are now automatically launched and controlled by the blahpd daemon so there is no need for a startup script.
The main differences between this approach and the old one are that the daemons run as non privileged user and that the informations about jobs are collected using the batch system native command (as bhist, bjobs, qstat...) and are saved in a db file on disk called job registry.
These informations are asynchronously notified to CREAM by BNotifier daemon.

It is possible to configure some parameter in the blah.config file:

- *bupdater_path* and *bnotifier_path*: if this is set to the bupdater (or bnotifier) complete path (e.g. /opt/glite/bin/BUpdaterLSF) blahpd takes care of starting and periodically check the status of the daemon. If it is not set the daemons should be started by hand.
- *bupdater_pidfile* and *bnotifier_pidfile*: this are used to check the daemons status by blahpd (e.g. /var/tmp/bupdater.pid).
- *job_registry*: is the complete path to the job registry location (/var/tmp/job_registry.db).
- *bupdater_debug_level*: set the debug level (1,2,3)
- *bupdater_debug_logfile*: set the location of the debug log file
- *async_notification_host*: hostname where the notification should be sent
- *async_notification_port*: port where to send notifications
- *purge_interval*: how old (in seconds) a job registry entry has to be before it is purged.
- *finalstate_query_interval*: the query that search for finished jobs is executed every finalstate_query_interval seconds (default:30)
- *alldone_interval*: after that interval an unseen job is set as done (status = 4) and exitstatus = -1 (default:600)
- *bupdater_loop_interval*: the updater main lop is executed every bupdater_loop_interval seconds (default:5)
- *blah_graceful_kill_mappable_cmd*: Command to use when terminating processes via a user mapping tool (glexec or SUDO). Default is /bin/kill
- *blah_graceful_kill_timeout*: Total number of seconds a parent process will wait for a child to terminate after sending it the first termination (SIGTERM) signal. During this time, and until the child process terminates, the parent will send SIGTERM to che child once a second before blah_graceful_kill_timeout/2 seconds have elapsed, then it will send SIGKILL once a second. Defaults to 20 seconds.
- *blah_children_restart_interval*: Mininum time in seconds that has to elapse before blahpd tries restarting one of the children processes it controls. Defaults to 150 seconds.
- *blah_require_proxy_on_submit*: Set to boolean 'true' to generate an error when the X509Proxy attribute is missing from the job description in the BLAH_JOB_SUBMIT command. Set to boolean 'false' if job submission without a valid proxy is allowed. Defaults to 'false'.
- *blah_enable_glexec_from_condor*: Set to boolean 'true' to make blahpd recognize the following three commands: CACHE_PROXY_FROM_FILE, USE_CACHED_PROXY, UNCACHE_PROXY. User mapping via GLEXEC will be automatically enabled when the USE_CACHED_PROXY command is received. This is to allow user switching via glexec based on the active proxy for a given user, as cached by the Condor gridmanager. Defaults to 'false'.
- *blah_disable_wn_proxy_renewal*: When set to boolean 'true', use of the BPRclient/server tool to delegate or send renewed X509 proxies to a job executing node will be entirely disabled. Defaults to 'false'.
- *blah_delegate_renewed_proxies*: When set to boolean 'true', X509 proxies sent to job worker nodes via the BPRclient/server tool will be delegated instead of being copied over the wire. Defaults to 'false' as many Globus tools (including gridftp) will reject access when a limited -and- delegated proxy is presented.

- *blah_graceful_kill_glexecable_cmd_format*

-- MassimoSgaravatto - 2011-12-21

This topic: CREAM > BlahUserGuide
Topic revision: r3 - 2013-02-21 - LisaZangrando