IBM Spectrum LSF for SAS
Version 10 Release 1.0

Command Reference

Contents

Chapter 1. bacct

Displays accounting statistics about finished jobs.

Synopsis

`bacct` [`-b` | `-l`[`-aff`] [`-gpu`]] [`-d`] [`-E`] [`-e`] [`-UF`] [`-w`] [`-x`] [`-cname`] [`-app` *application_profile_name*] [`-C` *time0*,*time1*] [`-D` *time0*,*time1*] [`-f` *logfile_name* | `-f -`] [`-Lp` *ls_project_name ...*] [`-m` *host_name ...*|`-M` *host_list_file*] [`-N` *host_name* | `-N` *host_model* | `-N` *cpu_factor*] [`-P` *project_name ...*] [`-q` *queue_name ...*] [`-sla` *service_class_name ...*] [`-S` *time0*,*time1*] [`-u` *user_name ...* | `-u all`] [`-f` *logfile_name*] [*job_ID ...*] [`-U` *resrvation_ID ...* | `-U all`]

`bacct` [`-h` | `-V`]

Description

Displays a summary of accounting statistics for all finished jobs (with a DONE or EXIT status) submitted by the user who ran the command, on all hosts, projects, and queues in the LSF system.

By default, the **bacct** command displays statistics for all jobs that are logged in the current LSF accounting log file: LSB_SHAREDIR/*cluster_name*/logdir/lsb.acct.

CPU time is not normalized.

All times are in seconds.

Statistics not reported by the **bacct** command but of interest to individual system administrators can be generated by directly using **awk** or **perl** to process the `lsb.acct` file.

Throughput calculations

To calculate the throughput (*T*) of the LSF system, specific hosts, or queues, use the following formula:

```
T = N/(ET-BT)
```

Where:

- *N* is the total number of jobs for which accounting statistics are reported
- *BT* is the job start time, when the first job was logged
- *ET* is the job end time, when the last job was logged

Use the option **-C** *time0*,*time1* to specify the job start time as *time0* and the end time as *time1*. In this way, you can examine throughput during a specific time period.

- To calculate the total throughput of the LSF system, specify the `-u all` option without any of the `-m`, `-q`, `-S`, `-D`, or *job_ID* options.
- To calculate the throughput of hosts, specify the **-u all** option *without* the `-q`, `-S`, `-D`, or *job_ID* options.
- To calculate the throughput of queues, specify the `-u all` option *without* the `-m`, `-S`, `-D`, or *job_ID* options.

Only the jobs that are involved in the throughput calculation are logged (that is, with a DONE or EXIT status). Jobs that are running, suspended, or that were never dispatched after submission are not considered, because they are still in the LSF system and not logged in the `lsb.acct` file.

The **bacct** command does not show local pending batch jobs that were killed with the **bkill -b** command. The **bacct** command shows LSF multicluster capability jobs and local running jobs even if they are killed by using the **bkill -b** command.

Options

-aff
Displays information about jobs with CPU and memory affinity resource requirement for each task in the job. A table headed AFFINITY shows detailed memory and CPU binding information for each task in the job, one line for each allocated processor unit.

Use only with the -l option.

-b
Brief format.

-E
Displays accounting statistics that are calculated with eligible pending time instead of total pending time for the wait time, turnaround time, expansion factor (turnaround time/run time), and hog factor (CPU time/turnaround time).

-d
Displays accounting statistics for successfully completed jobs (with a DONE status).

-e
Displays accounting statistics for exited jobs (with an EXIT status).

-gpu

bacct -l -gpu shows the following information on GPU job allocation after the job finishes:

Use this option only with the -l option.

Host Name
Name of the host.

GPU IDs on the host
Each GPU is shown as a separate line.

TASK and ID
List of job tasks and IDs using the GPU (separated by comma if used by multiple tasks)

MODEL
Contains the GPU brand name and model type name.

MTOTAL
The total GPU memory size.

GPU Compute Capability

MRSV
GPU memory reserved by the job

SOCKET
socket ID of the GPU located at

NVLINK
Indicates if the GPU has NVLink connections with other GPUs allocated for the job (ranked by GPU ID and including itself). The connection flag of each GPU is a character separated by "/" with the next GPU:

A "Y" indicates there is a direct NVLINK connection between two GPUs.

An "N" shows there is no direct NVLINK connection with that GPU.

A "-" shows the GPU is itself.

If the job exited abnormally due to a GPU-related error or warning, the error or warning message displays. If LSF could not get GPU usage information from DCGM, a hyphen (-) displays.

-l
Long format. Displays detailed information for each job in a multiline format.

If the job was submitted with the **bsub -K** command, the -l option displays Synchronous Execution.

-UF
Displays unformatted job detail information.

This option makes it easy to write scripts for parsing keywords on **bacct**. The results of this option have no wide control for the output. Each line starts from the beginning of the line. All lines that start with the time stamp are displayed unformatted in a single line. The output has no line length and format control.

-w
Wide field format.

-x
Displays jobs that triggered a job exception (overrun, underrun, idle, runtime_est_exceeded). Use with the **-l** option to show the exception status for individual jobs.

-cname
In IBM Spectrum LSF Advanced Edition, includes the cluster name for execution cluster hosts and host groups in output.

-app *application_profile_name*
Displays accounting information about jobs that are submitted to the specified application profile. You must specify an existing application profile that is configured in `lsb.applications`.

-C *time0,time1*
Displays accounting statistics for jobs that completed or exited during the specified time interval. Reads the `lsb.acct` file and all archived log files (`lsb.acct.`*n*) unless the `-f` option is used to specify a log file.

The time format is the same as in the **bhist** command.

-D *time0,time1*
Displays accounting statistics for jobs that are dispatched during the specified time interval. Reads the `lsb.acct` file and all archived log files (`lsb.acct.`*n*) unless the `-f` option is also used to specify a log file.

The time format is the same as in the **bhist** command.

-f *logfile_name* | -f -
Searches the specified job log file for accounting statistics, which is useful for offline analysis. Specify either an absolute or relative path.

The specified file path can contain up to 4094 characters for UNIX, or up to 512 characters for Windows.

Specify the `-f -` option to force the **bacct** command to use the `lsb.acct` log file for accounting statistics. If you are using IBM Spectrum LSF Explorer ("LSF Explorer") to load accounting log records, the `-f -` option (or any `-f` argument that specifies a log file) forces the **bacct** command to bypass LSF Explorer. For more details, refer to **LSF_QUERY_ES_SERVERS** and **LSF_QUERY_ES_FUNCTIONS** in the *IBM Spectrum LSF Configuration Reference*.

-Lp *ls_project_name ...*
Displays accounting statistics for jobs that belong to the specified LSF License Scheduler projects. If a list of projects is specified, project names must be separated by spaces and enclosed in quotation marks (") or (').

-M *host_list_file*
Displays accounting statistics for jobs that are dispatched to the hosts listed in a file (*host_list_file*) containing a list of hosts. The host list file has the following format:

- Multiple lines are supported
- Each line includes a list of hosts that are separated by spaces
- The length of each line must be fewer than 512 characters

-m *host_name ...*
Displays accounting statistics for jobs that are dispatched to the specified hosts.

If a list of hosts is specified, host names must be separated by spaces and enclosed in quotation marks (") or ('), and maximum length cannot exceed 1024 characters.

-N *host_name* | -N *host_model* | -N *cpu_factor*
Normalizes CPU time by the CPU factor of the specified host or host model, or by the specified CPU factor.

If you use the **bacct** command offline by indicating a job log file, you must specify a CPU factor.

-P *project_name ...*
Displays accounting statistics for jobs that belong to the specified projects. If a list of projects is specified, project names must be separated by spaces and enclosed in quotation marks (") or ('). You cannot use one double quotation mark (") and one single quotation mark (') to enclose the list.

-q *queue_name ...*
Displays accounting statistics for jobs that are submitted to the specified queues.

If a list of queues is specified, queue names must be separated by spaces and enclosed in quotation marks (") or (').

-S *time0,time1*
Displays accounting statistics for jobs that are submitted during the specified time interval. Reads the `lsb.acct` file and all archived log files (`lsb.acct.`*n*) unless the `-f` option is also used to specify a log file.

The time format is the same as in the **bhist** command.

-sla *service_class_name*
Displays accounting statistics for jobs that ran under the specified service class.

If a default system service class is configured with the **ENABLE_DEFAULT_EGO_SLA** parameter in the `lsb.params` file, but not explicitly configured in the `lsb.applications` file, the **bacct -sla** *service_class_name* command displays accounting information for the specified default service class.

-U *reservation_id ...* | -U all
Displays accounting statistics for the specified advance reservation IDs, or for all reservation IDs if the keyword all is specified.

A list of reservation IDs must be separated by spaces and enclosed in quotation marks (") or (').

The `-U` option also displays historical information about reservation modifications.

When combined with the `-U` option, the `-u` option is interpreted as the user name of the reservation creator. The following command shows all the advance reservations that are created by user `user2`.

```
bacct -U all -u user2
```

Without the `-u` option, the **bacct -U** command shows all advance reservation information about jobs that are submitted by the user.

In a LSF multicluster capability environment, advance reservation information is only logged in the execution cluster, so **bacct** displays advance reservation information for local reservations only. You cannot see information about remote reservations. You cannot specify a remote reservation ID, and the keyword all displays only information about reservations in the local cluster.

-u *user_name ...*|-u all
Displays accounting statistics for jobs that are submitted by the specified users, or by all users if the keyword `all` is specified.

If a list of users is specified, user names must be separated by spaces and enclosed in quotation marks (") or ('). You can specify both user names and user IDs in the list of users.

job_ID ...
Displays accounting statistics for jobs with the specified job IDs.

If the reserved job ID 0 is used, it is ignored.

In LSF multicluster capability job forwarding mode, you can use the local job ID and cluster name to retrieve the job details from the remote cluster.

General queries have the following syntax:

```
bacct submission_job_id@submission_cluster_name
```

Job arrays have the following query syntax:

```
bacct "submission_job_id[index]"@submission_cluster_name"
```

The advantage of using *submission_job_id*@*submission_cluster_name* instead of **bacct -l** *job_ID* is that you can use *submission_job_id*@*submission_cluster_name* as an alias to query a local job in the execution cluster without knowing the local job ID in the execution cluster. The **bacct** output is identical no matter which job ID you use (local job ID or *submission_job_id*@*submission_cluster_name*

You can use the **bacct 0** command to find all finished jobs in your local cluster, but **bacct 0@***submission_cluster_name* is not supported.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Default output format (SUMMARY)

Statistics on jobs. The following fields are displayed:

- Total number of done jobs
- Total number of exited jobs
- Total CPU time consumed
- Average CPU time consumed
- Maximum CPU time of a job
- Minimum CPU time of a job
- Total wait time in queues
- Average wait time in queue
- Maximum wait time in queue
- Minimum wait time in queue
- Average turnaround time (seconds/job)
- Maximum turnaround time
- Minimum turnaround time
- Average hog factor of a job (cpu time/turnaround time)
- Maximum hog factor of a job
- Minimum hog factor of a job
- Average expansion factor of a job (turnaround time/run time)
- Maximum expansion factor of a job
- Minimum expansion factor of a job
- Total run time consumed
- Average run time consumed
- Maximum run time of a job
- Minimum run time of a job

- Total throughput
- Beginning time is the completion or exit time of the first job selected
- Ending time is the completion or exit time of the last job selected

The total, average, minimum, and maximum statistics are on all specified jobs.

The wait time is the elapsed time from job submission to job dispatch.

The turnaround time is the elapsed time from job submission to job completion.

The hog factor is the amount of CPU time that is consumed by a job divided by its turnaround time.

The expansion factor of a job is its turnaround time divided by its run time.

The throughput is the number of completed jobs divided by the time period to finish these jobs (jobs/hour).

Output: Brief format (-b)

In addition to the default format SUMMARY, displays the following fields:

U/UID
Name of the user who submitted the job. If LSF fails to get the user name by **getpwuid**, the user ID is displayed.

QUEUE
Queue to which the job was submitted.

SUBMIT_TIME
Time when the job was submitted.

CPU_T
CPU time that is consumed by the job.

WAIT
Wait time of the job.

TURNAROUND
Turnaround time of the job.

FROM
Host from which the job was submitted.

EXEC_ON
Host or hosts to which the job was dispatched to run.

JOB_NAME
The job name that is assigned by the user, or the command string assigned by default at job submission with the **bsub** command. If the job name is too long to fit in this field, then only the latter part of the job name is displayed.

The displayed job name or job command can contain up to 4094 characters.

Output: Long format (-l)

Also displays host-based accounting information (CPU_T, MEM, and SWAP) for completed jobs when the **LSF_HPC_EXTENSIONS="HOST_RUSAGE"** parameter is set in the `lsf.conf` file.

In addition to the fields displayed by default in SUMMARY and by the `-b` option, displays the following fields:

JOBID
Identifier that LSF assigned to the job.

PROJECT_NAME
Project name that is assigned to the job.

STATUS
Status that indicates the job was either successfully completed (DONE status) or exited (EXIT status).

DISPATCH_TIME
Time when the job was dispatched to run on the execution hosts.

COMPL_TIME
Time when the job exited or completed.

HOG_FACTOR
Average hog factor, equal to *CPU_time / turnaround_time*.

MEM
Maximum resident memory usage of all processes in a job. By default, memory usage is shown in MB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for display (MB, GB, TB, PB, or EB).

CWD
Full path of the current working directory (CWD) for the job.

Specified CWD
User specified execution CWD.

SWAP
Maximum virtual memory usage of all processes in a job. By default, swap space is shown in MB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for display (MB, GB, TB, PB, or EB).

INPUT_FILE
File from which the job reads its standard input (see **bsub -i** *input_file*).

OUTPUT_FILE
File to which the job writes its standard output (see **bsub -o** *output_file*).

ERR_FILE
File in which the job stores its standard error output (see **bsub -e** *err_file*).

EXCEPTION STATUS
The exception status of a job includes the following possible values:

idle
The job is consuming less CPU time than expected. The job idle factor (*CPU_time*/*run_time*) is less than the configured **JOB_IDLE** threshold for the queue and a job exception was triggered.

overrun
The job is running longer than the number of minutes specified by the **JOB_OVERRUN** threshold for the queue and a job exception was triggered.

underrun
The job finished sooner than the number of minutes specified by the **JOB_UNDERRUN** threshold for the queue and a job exception was triggered.

runtime_est_exceeded
The job is running longer than the number of minutes specified by the runtime estimation and a job exception was triggered.

SYNCHRONOUS_EXECUTION
Job was submitted with the **-K** option. LSF submits the job and waits for the job to complete.

JOB_DESCRIPTION
The job description that is assigned by the user at job submission with **bsub**. This field is omitted if no job description was assigned.

The displayed job description can contain up to 4094 characters.

Dispatched <number> Tasks on Hosts
The number of tasks in the job and the hosts to which those tasks were sent for processing. Displayed when the if**LSB_ENABLE_HPC_ALLOCATION** parameter is set to Y or y in the `lsf.conf` file.

Allocated <number> Slot(s) on Host(s)
The number of slots that were allocated to the job based on the number of tasks, and the hosts on which the slots are allocated. Displayed when the **LSB_ENABLE_HPC_ALLOCATION** parameter is set to Y or y in the `lsf.conf` file.

Effective RES_REQ
Displays a job's effective resource requirement as seen by the scheduler after resolving any OR constructs.

PE Network ID
Displays network resource allocations for IBM Parallel Edition (PE) jobs that are submitted with the `bsub -network` option, or to a queue or an application profile with the **NETWORK_REQ** parameter defined.

```
bacct -l 210
Job <210>, User <user1>;, Project <default>, Status <DONE>. Queue <normal>,
                     Command <my_pe_job>
Tue Jul 17 06:10:28: Submitted from host <hostA>, CWD </home/pe_jobs>;
Tue Jul 17 06:10:31: Dispatched to <hostA>, Effective RES_REQ <select[type
                     == local] order[r15s:pg] rusage[mem=1.00] >, PE Network
                     ID <1111111>  <2222222> used <1> window(s)
                     per network per task;
Tue Jul 17 06:11:31: Completed <done>.
```

Output: Advance reservations (-U)

Displays the following fields:

RSVID
Advance reservation ID assigned by **brsvadd** command.

TYPE
Type of reservation: `user` or `system`.

CREATOR
User name of the advance reservation creator, who submitted the **brsvadd** command.

USER
User name of the advance reservation user, who submitted the job with the **bsub -U** command.

NCPUS
Number of CPUs reserved.

RSV_HOSTS
List of hosts for which processors are reserved, and the number of processors reserved.

TIME_WINDOW
Time window for the reservation.

- A one-time reservation displays fields that are separated by slashes (`month/day/hour/minute`).

  ```
  11/12/14/0-11/12/18/0
  ```

- A recurring reservation displays fields that are separated by colons (`day:hour:minute`).

  ```
  5:18:0 5:20:0
  ```

Output: Affinity resource requirements information (-l -aff)

Use the `-l -aff` option to display accounting job information about CPU and memory affinity resource allocations for job tasks. A table with the heading `AFFINITY` is displayed containing the detailed affinity information for each task, one line for each allocated processor unit. CPU binding and memory binding information are shown in separate columns in the display.

HOST
The host the task is running on.

TYPE
Requested processor unit type for CPU binding. One of `numa`, `socket`, `core`, or `thread`.

LEVEL
Requested processor unit binding level for CPU binding. One of `numa`, `socket`, `core`, or `thread`. If no CPU binding level is requested, a dash (`-`) is displayed.

EXCL
Requested processor unit binding level for exclusive CPU binding. One of `numa`, `socket`, `core`, or `thread`. If no exclusive binding level is requested, a dash (`-`) is displayed.

IDS
List of physical or logical IDs of the CPU allocation for the task.

The list consists of a set of paths, represented as a sequence of integers separated by slash characters (/), through the topology tree of the host. Each path identifies a unique processing unit that is allocated to the task. For example, a string of the form `3/0/5/12` represents an allocation to thread 12 in core 5 of socket 0 in NUMA node 3. A string of the form `2/1/4`represents an allocation to core 4 of socket 1 in NUMA node 2. The integers correspond to the node ID numbers displayed in the topology tree from **bhosts -aff**.

POL
Requested memory binding policy. Either`local` or `pref`. If no memory binding is requested, `-` is displayed.

NUMA
ID of the NUMA node that the task memory is bound to. If no memory binding is requested, a dash (`-`) is displayed.

SIZE
Amount of memory that is allocated for the task on the NUMA node.

For example, the following job starts 6 tasks with the following affinity resource requirements:

```
bsub -n 6 -R"span[hosts=1] rusage[mem=100]affinity[core(1,same=socket,
exclusive=(socket,injob)):cpubind=socket:membind=localonly:distribute=pack]" myjob
Job <6> is submitted to default queue <normal>.
```

```
bacct -l -aff 6

Accounting information about jobs that are:
  - submitted by all users.
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------------------------

Job <6>, User <user1>, Project <default>, Status <DONE>, Queue <normal>, Comma
                     nd <myjob>
Thu Feb 14 14:13:46: Submitted from host <hostA>, CWD <$HOME>;
Thu Feb 14 14:15:07: Dispatched 6 Task(s) on Host(s) <hostA> <hostA> <hostA>
                     <hostA> <hostA> <hostA>; Allocated <6> Slot(s) on Host(s)
                     <hostA> <hostA> <hostA> <hostA> <hostA> <hostA>;
                     Effective RES_REQ <select[type == local] order[r15s:pg]
                     rusage[mem=100.00] span[hosts=1] affinity
                     [core(1,same=socket,exclusive=(socket,injob))*1:cpubind=
                     socket:membind=localonly:distribute=pack] >
                     ;
Thu Feb 14 14:16:47: Completed <done>.

AFFINITY:
                    CPU BINDING                          MEMORY BINDING
                    -----------                          --------------
HOST                TYPE   LEVEL  EXCL   IDS            POL   NUMA SIZE
hostA               core   socket socket /0/0/0         local 0    16.7MB
hostA               core   socket socket /0/1/0         local 0    16.7MB
hostA               core   socket socket /0/2/0         local 0    16.7MB
hostA               core   socket socket /0/3/0         local 0    16.7MB
hostA               core   socket socket /0/4/0         local 0    16.7MB
hostA               core   socket socket /0/5/0         local 0    16.7MB

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.01       81            181     done         0.0001     2M    137M
------------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done jobs:       1      Total number of exited jobs:     0
 Total CPU time consumed:       0.0      Average CPU time consumed:     0.0
```

```
 Maximum CPU time of a job:     0.0      Minimum CPU time of a job:     0.0
 Total wait time in queues:    81.0
 Average wait time in queue:   81.0
 Maximum wait time in queue:   81.0      Minimum wait time in queue:   81.0
 Average turnaround time:       181 (seconds/job)
 Maximum turnaround time:       181      Minimum turnaround time:       181
 Average hog factor of a job:  0.00 ( cpu time / turnaround time )
 Maximum hog factor of a job:  0.00      Minimum hog factor of a job:  0.00
 Average expansion factor of a job:  1.00 (turnaround time/run time)
 Maximum expansion factor of a job:  1.00   Minimum expansion factor of a job:  1.00
...
```

Termination reasons displayed by bacct

When LSF detects that a job is terminated, **bacct -l** displays one of the following termination reasons. The corresponding exit code integer value that is logged to the JOB_FINISH record in the lsb.acct file is given in parentheses.

- TERM_ADMIN: Job was killed by root or LSF administrator (15)
- TERM_BUCKET_KILL: Job was killed with the **bkill -b** command (23)
- TERM_CHKPNT: Job was killed after checkpointing (13)
- TERM_CWD_NOTEXIST: current working directory is not accessible or does not exist on the execution host (25)
- TERM_CPULIMIT: Job was killed after it reached LSF CPU usage limit (12)
- TERM_DEADLINE: Job was killed after deadline expires (6)
- TERM_EXTERNAL_SIGNAL: Job was killed by a signal external to LSF (17)
- TERM_FORCE_ADMIN: Job was killed by root or LSF administrator without time for cleanup (9)
- TERM_FORCE_OWNER: Job was killed by owner without time for cleanup (8)
- TERM_LOAD: Job was killed after load exceeds threshold (3)
- TERM_MEMLIMIT: Job was killed after it reached LSF memory usage limit (16)
- TERM_ORPHAN_SYSTEM: The orphan job was automatically terminated by LSF (27)
- TERM_OWNER: Job was killed by owner (14)
- TERM_PREEMPT: Job was killed after preemption (1)
- TERM_PROCESSLIMIT: Job was killed after it reached LSF process limit (7)
- TERM_REMOVE_HUNG_JOB: Job was removed from LSF system after it reached a job runtime limit (26)
- TERM_REQUEUE_ADMIN: Job was killed and requeued by root or LSF administrator (11)
- TERM_REQUEUE_OWNER: Job was killed and requeued by owner (10)
- TERM_RUNLIMIT: Job was killed after it reached LSF runtime limit (5)
- TERM_SWAP: Job was killed after it reached LSF swap usage limit (20)
- TERM_THREADLIMIT: Job was killed after it reached LSF thread limit (21)
- TERM_UNKNOWN: LSF cannot determine a termination reason. 0 is logged but TERM_UNKNOWN is not displayed (0)
- TERM_WINDOW: Job was killed after queue run window closed (2)
- TERM_ZOMBIE: Job exited while LSF is not available (19)

Tip: The integer values logged to the JOB_FINISH record in the lsb.acct file and termination reason keywords are mapped in the lsbatch.h header file.

Example: Default format

```
bacct
Accounting information about jobs that are:
  - submitted by users user1.
  - accounted on all projects.
  - completed normally or exited.
```

```
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------ -----

SUMMARY:      ( time unit: second )
Total number of done jobs:     268      Total number of exited jobs:    31
Total CPU time consumed:     566.4      Average CPU time consumed:     1.9
Maximum CPU time of a job:   229.9      Minimum CPU time of a job:     0.0
Total wait time in queues:   393.0
Average wait time in queue:    1.3
Maximum wait time in queue:   97.0      Minimum wait time in queue:    0.0
Average turnaround time:        32 (seconds/job)
Maximum turnaround time:       301      Minimum turnaround time:         0
Average hog factor of a job:  0.16 ( cpu time / turnaround time )
Maximum hog factor of a job:  0.91      Minimum hog factor of a job:  0.00
 Average expansion factor of a job:  1.13 (turnaround time/run time)
 Maximum expansion factor of a job:  2.04   Minimum expansion factor of a job:  1.00
Total Run time consumed:      9466      Average Run time consumed:      31
 Maximum Run time of a job:     300      Minimum Run time of a job:       0
Total throughput:           122.17 (jobs/hour)  during    2.45 hours
Beginning time:       Oct 20 13:40      Ending time:          Oct 20 16:07
```

Example: Jobs with triggered job exceptions

```
bacct -x -l

Accounting information about jobs that are:
  - submitted by users user1,
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------

Job <1743>, User <user1>, Project <default>, Status <DONE>, Queue <normal>,  Command<sleep 30>
Mon Aug 11 18:16:17 2009: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File </dev/null>;
Mon Aug 11 18:17:22 2009: Dispatched to <hostC>; Effective RES_REQ <select[(hname = delgpu3 ) &&
                                          (type == any)] order[r15s:pg]>;
Mon Aug 11 18:18:54 2009: Completed <done>.


 EXCEPTION STATUS:  underrun

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.19       65            157     done         0.0012     4M      5M
------------------------------------------------------------ --------
Job <1948>, User <user1>, Project <default>, Status <DONE>, Queue <normal>,Command <sleep 550>,
Job Description <This job is a test job.>
Tue Aug 12 14:15:03 2009: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File </dev/null>;
Tue Aug 12 14:15:15 2009: Dispatched to <hostC>; Effective RES_REQ <select[(hname = delgpu3 ) &&
                                          (type == any)] order[r15s:pg]>;
Tue Aug 12 14:25:08 2009: Completed <done>.


 EXCEPTION STATUS:  overrun  idle

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.20       12            605     done         0.0003     4M      5M
------------------------------------------------------------
Job <1949>, User <user1>, Project <default>, Status <DONE>, Queue <normal>,Command <sleep 400>
Tue Aug 12 14:26:11 2009: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File </dev/null>;
Tue Aug 12 14:26:18 2009: Dispatched to <hostC>; Effective RES_REQ <select[(hname = delgpu3 )
                                          && (type == any)] order[r15s:pg]>;
Tue Aug 12 14:33:16 2009: Completed <done>.


 EXCEPTION STATUS:  idle

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
```

```
      0.17        7            425     done         0.0004     4M      5M
```

```
Job <719[14]>, Job Name <test[14]>, User <user1>, Project <default>, Status <EXIT>, Queue <normal>,
Command </home/user1/job1>, Job Description <This job is another test job.>
Mon Aug 18 20:27:44 2009: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File </dev/null>;
Mon Aug 18 20:31:16 2009: [14] dispatched to <hostA>; Effective RES_REQ <select[(hname = delgpu3 )
                                          && (type == any)] order[r15s:pg]>;
Mon Aug 18 20:31:18 2009: Completed <exit>.
```

```
 EXCEPTION STATUS:  underrun

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.19      212            214     exit         0.0009     2M      4M
------------------------------------------------------------ -----------
```

```
SUMMARY:      ( time unit: second )
 Total number of done jobs:      45      Total number of exited jobs:    56
 Total CPU time consumed:    1009.1      Average CPU time consumed:    10.0
 Maximum CPU time of a job:   991.4      Minimum CPU time of a job:     0.1
 Total wait time in queues: 116864.0
 Average wait time in queue: 1157.1
 Maximum wait time in queue: 7069.0      Minimum wait time in queue:    7.0
 Average turnaround time:      1317 (seconds/job)
 Maximum turnaround time:      7070      Minimum turnaround time:        10
 Average hog factor of a job:  0.01 ( cpu time / turnaround time )
 Maximum hog factor of a job:  0.56      Minimum hog factor of a job:  0.00
 Average expansion factor of a job: 4.6 (turnaround time/run time)
 Maximum expansion factor of a job: 10.2 Minimum expansion factor of a job: 1.00
 Total Run time consumed:     28987      Average Run time consumed:     287
 Maximum Run time of a job:    6743      Minimum Run time of a job:       2
 Total throughput:             0.59 (jobs/hour)  during  170.21 hours
 Beginning time:       Aug 11 18:18      Ending time:          Aug 18 20:31
```

Example: Advance reservation accounting information

```
bacct -U user1#2
Accounting for:
  - advance reservation IDs: user1#2
  - advance reservations created by user1
------------------------------------------------------------ -----------
RSVID       TYPE      CREATOR    USER    NCPUS       RSV_HOSTS     TIME_WINDOW
user1#2     user        user1   user1      1           hostA:1    9/16/17/36-9/16/17/38
SUMMARY:
Total number of jobs:               4
Total CPU time consumed:      0.5 second
Maximum memory of a job:     4.2 MB
Maximum swap of a job:         5.2 MB
Total duration time:                0 hour    2 minute    0 second
```

Example: LSF job termination reason logging

When a job finishes, LSF reports the last job termination action that it took against the job and logs it to the `lsb.acct` file.

If a running job exits because of node failure, LSF sets the correct exit information in the `lsb.acct`, `lsb.events`, and the job output file.

Use the **bacct -l** command to view job exit information that is logged to the `lsb.acct` file:

```
bacct -l 7265

Accounting information about jobs that are:
  - submitted by all users.
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
```

```
------------------------------------------------------------ ---------
```

```
Job <7265>, User <lsfadmin>, Project <default>, Status <EXIT>, Queue <normal>, Command
<srun sleep 100000>, Job Description <This job is also a test job.>
Thu Sep 16 15:22:09 2009: Submitted from host <hostA>, CWD <$HOME>;
Thu Sep 16 15:22:20 2009: Dispatched to 4 Hosts/Processors <4*hostA>;
Thu Sep 16 15:23:21 2009: Completed <exit>; TERM_RUNLIMIT: job killed after reaching LSF run time limit.

Accounting information about this job:
     Share group charged </lsfadmin>
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR    MEM    SWAP
      0.04       11             72     exit         0.0006     0K      0K
------------------------------------------------------------ ----------
```

```
SUMMARY:      ( time unit: second )
 Total number of done jobs:       0      Total number of exited jobs:     1
 Total CPU time consumed:       0.0      Average CPU time consumed:     0.0
 Maximum CPU time of a job:     0.0      Minimum CPU time of a job:     0.0
 Total wait time in queues:    11.0
 Average wait time in queue:   11.0
 Maximum wait time in queue:   11.0      Minimum wait time in queue:   11.0
 Average turnaround time:        72     (seconds/job)
 Maximum turnaround time:        72      Minimum turnaround time:        72
 Average hog factor of a job:  0.00     (cpu time / turnaround time )
 Maximum hog factor of a job:  0.00      Minimum hog factor of a job:  0.00

 ...
```

Example: Resizable job information

Use the **bacct -l** command to view resizable job information that is logged to the `lsb.acct` file:

- The autoresizable attribute of a job and the resize notification command if the **bsub -ar** and **bsub -rnc** *resize_notification_command* commands are specified.
- Job allocation changes whenever a JOB_RESIZE event is logged to the `lsb.acct` file.

When an allocation grows, the **bacct** command shows

```
Additional allocation on <num_hosts> Hosts/Processors <host_list>
```

When an allocation shrinks, the **bacct** command shows

```
Release allocation on <num_hosts> Hosts/Processors <host_list> by user or
administrator <user_name>
Resize notification accepted;
```

For the following job is submission:

```
bsub -n 1, 5 -ar myjob
```

The initial allocation is on hostA and hostB. The first resize request is allocated on hostC and hostD. A second resize request is allocated on hostE. The **bacct -l** command has the following output:

```
bacct -l 205

Accounting information about jobs that are:
  - submitted by all users.
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------------------------
```

```
Job <1150>, User <user2>, Project <default>, Status <DONE>, Queue <normal>, Command
<sleep 10>, Job Description <This job is a test job.>
Mon Jun  2 11:42:00 2009: Submitted from host <hostA>, CWD <$HOME>;
Mon Jun  2 11:43:00 2009: Dispatched 6 Task(s) on Host(s) <hostA> <hostB>,
```

```
                         Allocated 6 Slot(s) on Host(s) <hostA> <hostB>,
                         Effective RES_REQ <select[(hname = delgpu3 ) &&
                         (type == any)] order[r15s:pg]>;
```

```
Mon Jun  2 11:43:52 2009: Added 2 Task(s) on Host(s) 2 Hosts/Processors
                         <hostC> <hostD>, 2 additional Slot(s) allocated
                         on Host(s) <hostC> <hostD>
Mon Jun  2 11:44:55 2009: Additional allocation on <hostC> <hostD>;
Mon Jun  2 11:51:40 2009: Completed <done>.
...
```

Files

Reads the `lsb.acct` and `lsb.acct.`*n* files.

See also

bhist, **bsub**, **bjobs**, `lsb.acct`, **brsvadd**, **brsvs**, **bsla**, `lsb.serviceclasses`

Chapter 2. badmin

The **badmin** command is the administrative tool for LSF.

Synopsis

`badmin` *subcommand options*

`badmin [-h | -V]`

Description

The **badmin** command provides a set of subcommands to control and monitor LSF. If you do not include subcommands, the `badmin` command prompts for subcommands from the standard input.

Information about each subcommand is available through the `-h` option.

The **badmin** subcommands include privileged and non-privileged subcommands. Only root or LSF administrators can run privileged subcommands. The following subcommands are privileged:

diagnose
gpddebug
gpdrestart
gpdtime
hclose
hghostadd
hghostdel
hopen
hpower
hrestart
hshutdown
hstartup
mbddebug
mbdrestart
perflog
perfmon
qact
qclose
qinact
qopen
rc
reconfig

The configuration file `lsf.sudoers` must be set to use the privileged command **hstartup** by a non-root user.

All other commands are non-privileged commands and can be used by any LSF user. If the **LSF_AUTH** parameter is not defined in the `lsf.conf` file, privileged ports are used and the **badmin** command must be installed because it needs to send the request through a privileged port. The **badmin** executable file is installed with the setuid flag turned on.

When you use subcommands for which multiple host names can be specified, do not enclose the host names in quotation marks.

Subcommand synopsis

`ckconfig [-v]`

`diagnose` *pending_jobID* ...

`diagnose -c jobreq` [`-f` *logfile_name*] [`-t xml` | `json`]

`diagnose -c query` [[`-f` *logfile_name*] [`-d` *duration*] | [`-o`]]

`gpdckconfig` [`-v`]

`gpddebug` [`-c` *class_name*] [`-l` *debug_level*] [`-f` *logfile_name*] [`-o`]

`gpdrestart` [`-v`] [`-f`]

`gpdtime` [`-l` *timing_level*] [`-f` *logfile_name*] [`-o`]

`hclose` [`-C` *comment*] [*host_name* ... | *host_group* ... | *compute_unit* ... | `all`]

`help` [*command* ...] | `?` [*command* ...]

`hghostadd` [`-C` *comment*] *host_group* | *compute_unit* | *host_name* [*host_name* ...]

`hghostdel` [`-f`] [`-C` *comment*] *host_group* | *compute_unit* | *host_name* [*host_name* ...]

`hhist` [`-t` *time0*`,`*time1*] [`-f` *logfile_name*] [*host_name* ...]

`hist` [`-t` *time0*`,`*time1*] [`-f` *logfile_name*]

`hopen` [`-C` *comment*] [*host_name* ... | *host_group* ... | *compute_unit* ... | `all`]

`hpower` [`suspend` | `resume`] [`-C` *comment*] [*host_name* ...]

`hrestart` [`-f`] [*host_name* ... | `all`]

`hshutdown` [`-f`] [*host_name* ... | `all`]

`hstartup` [`-f`] [*host_name* ... | `all`]

`mbddebug` [`-c` *class_name* ...] [`-l` *debug_level*] [`-f` *logfile_name*] [`-o`] [`-s` *log_queue_size*]

`mbdhist` [`-t` *time0*`,`*time1*] [`-f` *logfile_name*]

`mbdrestart` [`-C` *comment*] [`-v`] [`-f`] [`-p` | `-s`]

`mbdtime` [`-l` *timing_level*] [`-f` *logfile_name*] [`-o`]

`perflog` [`-t` *sample_period*] [`-d` *duration*] [`-f` *logfile_name*] [`-o`]

`perfmon start` [*sample_period*]| `stop` | `view` | `setperiod` *sample_period*

`qact` [`-C` *comment*] [*queue_name* ... | `all`]

`qclose` [`-C` *comment*] [*queue_name* ... | `all`]

`qhist` [`-t` *time0*`,`*time1*] [`-f` *logfile_name*] [*queue_name* ...]

`qinact` [`-C` *comment*] [*queue_name* ... | `all`]

`qopen` [`-C` *comment*] [*queue_name* ... | `all`]

`quit`

`rc error` [`-t` *days*`d` | *hours*`h` | *minutes*`m`] [`-p "` *provider* ..."]

`rc view` [`-c "instances` | `policies` | `templates` ..."] [`-p "` *provider* ..."]

`reconfig` [`-v`] [`-f`]

`sbddebug` [`-c` *class_name* ...] [`-l` *debug_level*] [`-f` *logfile_name*] [`-o`] [*host_name* ...]

`sbdtime` [`-l` *timing_level*] [`-f` *logfile_name*] [`-o`] [*host_name* ...]

`schddebug` [`-c` *class_name* ...] [`-l` *debug_level*] [`-f` *logfile_name*] [`-o`] [`-s` *log_queue_size*]

`schdtime` [`-l` *timing_level*] [`-f` *logfile_name*] [`-o`]

`showconf mbd` | [`sbd` [*host_name* ... | `all`] | `gpd`]

`showstatus`

`-h`

`-V`

Options

subcommand
Runs the specified subcommand. See the Usage section.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Usage

ckconfig [-v]
Checks LSF configuration files that are located in the `LSB_CONFDIR/cluster_name/configdir` directory, and checks the `LSF_ENVDIR/lsf.licensescheduler` file.

The **LSB_CONFDIR** variable is defined in the `lsf.conf` file, in **LSF_ENVDIR** or `/etc` (if **LSF_ENVDIR** is not defined).

By default, the **badmin ckconfig** command displays only the result of the configuration file check. If warning errors are found, the **badmin** command prompts you to display detailed messages.

-v
Verbose mode. Displays detailed messages about configuration file checking to `stderr`.

diagnose *<pend jobid>* ...
Displays full pending reason list if **CONDENSE_PENDING_REASONS=Y** is set in the `lsb.params` file.

```
badmin diagnose 1057
```

diagnose -c jobreq [-f *snapshot_file_name*] [-t xml | json]
UNIX only. Saves the current contents of the scheduler job bucket information into an XML or JSON snapshot file as raw data.

Jobs are put into scheduling buckets based on resource requirements and different scheduling policies. Saving the contents into a snapshot file is useful for data analysis by parsing the file or by running a simple text search on its contents.

This feature is helpful if you want to examine a sudden large performance impact on the scheduler. Use the snapshot file to identify any users with many buckets or large attribute values.

You can use the following options:

-c jobreq
Required.

-f *file_name*
Specifies a snapshot file in which to save the information. It is either a file name, which is located in the **DIAGNOSE_LOGDIR** directory, or a full path file name. If the specified snapshot file exists, it is overwritten with the current information.

The default name for the snapshot file is `jobreq_<hostname>_<dateandtime>.<format>`, where `<format>` is `xml` or `json`, depending on the specified format of the snapshot file.

The owner of the log file is the user who is specified in the **LSF_ADMIN** parameter. The log file permissions are the same as the **mbatchd** daemon log permissions. Everyone has read and execute access but the **LSF_ADMIN** owner has write, read, and execute access.

-t xml | json
Specifies the format of the snapshot file. Specify `-t xml` for the snapshot file to be in XML format, or specify `-t json` for the snapshot file to be in JSON format.

The default format for the snapshot file is XML, and the extension of the snapshot file is `.xml`. If the snapshot file is in JSON format, the extension of the snapshot file is `.json`.

diagnose -c query [-f *logfile_name*] [-d *minutes*] | [-o]]

This feature is helpful if an unexpected **mbatchd** query load causes the cluster to slow or fail to respond to requests. For example, many **bjobs** command queries might cause a high network load and prevent the **mbatchd** daemon from responding. Running this command with its options enables the **mbatchd** daemon to dump the query source information into a log file.

The log file shows information about the source of queries for easier troubleshooting. The log file shows who made these requests, where the requests came from, and the data size of the query.

You can also configure this feature by enabling the **DIAGNOSE_LOGDIR** and **ENABLE_DIAGNOSE** parameters in the `lsb.params` file to log the entire query information as soon as the cluster starts. However, the dynamic settings from the command override the static parameter settings. Also, after the duration you specify to track the query information expires, the static diagnosis settings take effect.

You can use the following options to dynamically set the time, specify a log file, and allow the **mbatchd** daemon to collect information:

-c query

Required.

-f

Specifies a log file in which to save the information. It is either a file name, which is located in the **DIAGNOSE_LOGDIR** directory, or a full path file name.

The default name for the log file is `query_info.querylog.`<*host_name*>.

The owner of the log file is the user who is specified in the **LSF_ADMIN** parameter. The log file permissions are the same as **mbatchd** daemon log permissions. Everyone has read and execute access but the **LSF_ADMIN** user has write, read, and execute access.

If you specify the log file in the `lsb.params` file and then later specify a different log file in the command line, the one in the command line takes precedence. Logging continues until the specified duration is over, or until you stop dynamic logging. It then switches back to the static log file location.

-d *minutes*

The duration in minutes you specify to track the query information. The **mbatchd** daemon reverts to static settings after the duration is over, or until you stop it manually, restart (with the **badmin mbdrestart** command), or reconfigure (with **badmin reconfig** command). The default value for this duration is infinite. By default, query information is always logged.

-o

Turns off dynamic diagnosis (stop logging). If the **ENABLE_DIAGNOSE=query** parameter is configured, it returns to the static configuration.

gpdckconfig [-v]

Checks the global policy configuration file `lsb.globalpolicies` located in the `LSB_CONFDIR/cluster_name/configdir` directory.

The **LSB_CONFDIR** variable is defined in the `lsf.conf` file, in **LSF_ENVDIR** or `/etc` (if **LSF_ENVDIR** is not defined).

By default, the **badmin gpdckconfig** command displays only the result of the configuration file check. If warning errors are found, the **badmin** command prompts you to display detailed messages.

You can run the **badmin gpdckconfig** command only on the master host or master candidate hosts in the Global Policy Daemon Cluster (GPD Cluster).

-v

Verbose mode. Displays detailed messages about configuration file checking to `stderr`.

gpddebug [-c *class_name ...*] [-l *debug_level*] [-f *logfile_name*] [-o]

Sets the message log level for the **gpolicyd** daemon to include additional information in log files. You must be `root` or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

class_name
Not defined (no additional classes are logged).

***debug_level*=0**
As specified but the LOG_DEBUG level in the **LSF_LOG_MASK** parameter.

logfile_name
Not defined (LSF system log file in the LSF system log file directory, in the format `gpolicyd.log.`*`host_name`*).

-c *class_name ...*
Specifies software classes for which debug messages are to be logged.

By default, *class_name* is not defined and no additional classes are logged.

The format of *class_name* is the name of a class, or a list of class names separated by spaces and enclosed in quotation marks. Classes are also listed in the `lsf.h` header file.

The following log classes are supported:

LC_AUTH
Log authentication messages.

LC_COMM
Log communication messages.

LC_SYS
Log system call messages.

LC_TRACE
Log significant program walk steps.

LC_XDR
Log everything that is transferred by XDR.

LC_XDRVERSION
Log messages for XDR version.

LC2_G_FAIR
Log global fairshare messages.

-l *debug_level*
Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower levels.

debug_level has the following values:

Default: 0
LOG_DEBUG level in parameter **LSF_LOG_MASK**.

0
LOG_DEBUG level for parameter **LSF_LOG_MASK** in the `lsf.conf` file.

1
LOG_DEBUG1 level for extended logging. A higher level includes lower logging levels. For example, the **LOG_DEBUG1** level includes the **LOG_DEBUG** level.

2
LOG_DEBUG2 level for extended logging. A higher level includes lower logging levels. For example, the **LOG_DEBUG2** level includes **LOG_DEBUG1** and **LOG_DEBUG** levels.

3
LOG_DEBUG3 level for extended logging. A higher level includes lower logging levels. For example, the **LOG_DEBUG3** level includes **LOG_DEBUG2**, **LOG_DEBUG1**, and **LOG_DEBUG** levels.

-f *logfile_name*
Specifies the name of the file into which debugging messages are to be logged. A file name with or without a full path can be specified.

If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file that is created has the following format:

logfile_name.gpolicyd.log.*host_name*

On UNIX, if the specified path is not valid, the log file is created in the /tmp directory.

On Windows, if the specified path is not valid, no log file is created.

By default, *logfile_name* is the current LSF system log file in the LSF system log file directory.

-o
Turns off temporary debug settings and resets them to the daemon start state. The message log level is reset back to the value of **LSF_LOG_MASK** and classes are reset to the value of **LSB_DEBUG_GPD**.

The log file is also reset back to the default log file.

gpdrestart [-v] [-f]
Dynamically reconfigures LSF global policies and restarts the **gpolicyd** daemon.

The global policy configuration file lsb.globalpolicies is checked for errors and the results are printed to stderr. If no errors are found, the lsb.globalpolicies file is reloaded and the **gpolicyd** daemon is restarted.

If warning errors are found, the **badmin** command prompts you to display detailed messages. If unrecoverable errors are found, the **gpolicyd** daemon is not restarted, and the **badmin** command exits.

You can run the **badmin gpdrestart** command only on the master host or master candidate hosts in the Global Policy Daemon Cluster (GPD Cluster).

-v
Verbose mode. Displays detailed messages about the status of configuration files. All messages from configuration checking are printed to stderr.

-f
Disables interaction and proceeds with the **gpolicyd** daemon restart if configuration files contain no unrecoverable errors.

gpdtime [-l *timing_level*] [-f *logfile_name*] [-o]
Sets the timing level for the **gpolicyd** daemon to include extra timing information in log files. You must be root or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

timing_level
Not defined (timing information is recorded).

logfile_name
Not defined (current LSF system log file in the LSF system log file directory, in the format gpolicyd.log.*host_name*).

-l *timing_level*
Specifies the detail of timing information that is included in log files. Timing messages indicate the execution time of functions in the software and are logged in milliseconds.

The following values are supported: 1 | 2 | 3 | 4 | 5

The higher the number, the more functions in the software that are timed and whose execution time is logged. The lower numbers include more common software functions. Higher levels include all lower levels.

By default no timing information is logged.

-f *logfile_name*
Specify the name of the file into which timing messages are to be logged. A file name with or without a full path can be specified.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file that is created has the following format:

```
logfile_name.gpolicyd.log.host_name
```

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

Note: Both timing and debug messages are logged in the same files.

The default is the current LSF system log file in the LSF system log file directory, in the format `gpolicyd.log.host_name`.

-o
Optional. Turns off temporary timing settings and resets them to the daemon start state. The timing level is reset back to the value of the parameter for the corresponding daemon (**LSB_TIME_GPD**).

The log file is also reset back to the default log file.

hclose [-C *comment*] [*host_name* ... | *host_group* ... | *compute_unit* ... | all]
Closes batch server hosts. Specify the names of any server hosts, host groups, or compute units. All batch server hosts are closed if the reserved word `all` is specified. If no argument is specified, the local host is assumed. A closed host does not accept any new jobs, but jobs that are already dispatched to the host are not affected. This behavior is different from a host closed by a window; all jobs on a host are suspended when a time window closes on the host.

-C *comment*
Logs the text as an administrator comment record to the `lsb.events` file. The maximum length of the comment string is 512 characters.

If you close a host group or compute unit, each member is displayed with the same comment string.

You cannot use the **badmin hopen** command to open a host that was borrowed through LSF resource connector that is in `closed_RC` status.

help [*command* ...] | ? [*command* ...]
Displays the syntax and functions of the specified commands.

hghostadd [-C *comment*] *host_group* | *compute_unit* |*host_name* [*host_name* ...]
If dynamic host configuration is enabled, dynamically adds hosts to a host group or compute unit. After the **mbatchd** daemon receives the host information from the master LIM, it dynamically adds the host without triggering reconfiguration.

After the host is added to the host group or compute unit, it is considered part of that group for scheduling decisions for newly submitted jobs and for existing pending jobs.

This command fails if any of the specified host groups, compute units, or host names are not valid.

Restriction: If EGO-enabled SLA scheduling is configured through the **ENABLE_DEFAULT_EGO_SLA** parameter in the `lsb.params` file, you cannot use the **hghostadd** subcommand because all host allocation is under control of enterprise grid orchestrator (EGO).

-C *comment*
Logs the text as an administrator comment record to the `lsb.events` file. The maximum length of the comment string is 512 characters.

hghostdel [-f] [-C *comment*] *host_group* | *compute_unit* |*host_name* [*host_name* ...]
Dynamically deletes hosts from a host group or compute unit by triggering reconfiguration of the **mbatchd** daemon.

This command fails if any of the specified host groups, compute units, or host names are not valid.

CAUTION:

To change a dynamic host to a static host, first use the command **badmin hghostdel** to remove the dynamic host from any host group or compute unit that it belongs to. Then, configure the host as a static host in the `lsf.cluster.`*`cluster_name`* file.

Restriction: If EGO-enabled SLA scheduling is configured through the **ENABLE_DEFAULT_EGO_SLA** parameter in the `lsb.params` file, you cannot use the **hghostdel** subcommand because all host allocation is under control of enterprise grid orchestrator (EGO).

-f
Disables interaction and does not ask for confirmation when reconfiguring **mbatchd**.

-C *comment*
Logs the text as an administrator comment record to the `lsb.events` file. The maximum length of the comment string is 512 characters.

hhist [-t *time0*,*time1*] [-f *logfile_name*] [*host_name* ...]
Displays historical events for specified hosts, or for all hosts if no host is specified. Host events are host opening and closing. Also, both **badmin** command and policy- or job triggered power-related events (suspend, resume, reset) are displayed.

-t *time0*,*time1*

Displays only those events that occurred during the period from *time0* to *time1*. See the **bhist** command for the time format. The default is to display all host events in the event log file.

-f *logfile_name*
Specify the file name of the event log file. Either an absolute or a relative path name can be specified. The default is to use the current event log file in the LSF system: `LSB_SHAREDIR/`*`cluster_name`*`/logdir/lsb.events`. Option `-f` is useful for offline analysis.

If you specified an administrator comment with the `-C` option of the host control commands **hclose** or **hopen**, **hhist** displays the comment text.

hist [-t *time0*,*time1*] [-f *logfile_name*]
Displays historical events for all the queues, hosts, and **mbatchd**. Both **badmin** command and policy- or job-triggered power-related events (suspend, resume, reset) are displayed.

-t *time0*,*time1*
Displays only those events that occurred during the period from *time0* to *time1*. See **bhist** for the time format. The default is to display all queue events in the event log file.

-f *logfile_name*
Specify the file name of the event log file. Either an absolute or a relative path name can be specified. The default is to use the current event log file in the LSF system: `LSB_SHAREDIR/`*`cluster_name`*`/logdir/lsb.events` file. Option `-f` is useful for offline analysis.

If you specified an administrator comment with the `-C` option of the queue, host, and **mbatchd** daemon commands, the `hist` option displays the comment text.

hopen [-C *comment*] [*host_name* ... | *host_group*] [*host_name* ... | *host_group* ... | *compute_unit* ... | all]
Opens batch server hosts. Specify the names of any server hosts, host groups, or compute units. All batch server hosts are opened if the reserved word `all` is specified. If no host, host group, or compute unit is specified, the local host is assumed. A host accepts batch jobs if it is open.

Important: If EGO-enabled SLA scheduling is configured through the **ENABLE_DEFAULT_EGO_SLA** parameter in the `lsb.params`file, and a host is closed by EGO, it cannot be reopened by the **badmin hopen** command. Hosts closed by EGO have status `closed_EGO` in the **bhosts -l** command output.

-C *comment*
Logs the text as an administrator comment record to the `lsb.events` file. The maximum length of the comment string is 512 characters.

If you open a host group or compute unit, each member is displayed with the same comment string.

hpower [suspend | resume] [-C *comment*] [hostname...]

Manually switches hosts between a power-saving state or a working state.

suspend | resume

The state that you want to switch the host to.

-C *comment*

Logs the text as an administrator comment record to the `lsb.events` file. The maximum length of the comment string is 512 characters.

hrestart [-f] [*host_name* ... | all]

Restarts the **sbatchd** daemon on the specified hosts, or on all server hosts if the reserved word `all` is specified. If no host is specified, the local host is assumed. The **sbatchd** daemon reruns itself from the beginning. This rerun allows new **sbatchd** binary files to be used.

-f

Specify the name of the file into which timing messages are to be logged. A file name with or without a full path can be specified.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file has the following format:

```
logfile_name.daemon_name.log.host_name
```

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

Note: Both timing and debug messages are logged in the same files.

The default is the current LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*.

hshutdown [-f] [*host_name* ... | all]

Shuts down the **sbatchd** daemon on the specified hosts, or on all batch server hosts if the reserved word `all` is specified. If no host is specified, the local host is assumed. The **sbatchd** daemon exits after it receives the request.

-f

Disables interaction and does not ask for confirmation for shutting down **sbatchd**.

hstartup [-f] [*host_name* ... | all]

Starts the **sbatchd** daemon on the specified hosts, or on all batch server hosts if the reserved word `all` is specified. Only `root` and users who are listed in the `lsf.sudoers` file can use the `all` and `-f` options. If no host is specified, the local host is assumed.

-f

Disables interaction and does not ask for confirmation for starting the **sbatchd** daemon.

mbddebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o] [-s *log_queue_size*]

Sets message log level for the **mbatchd** daemon to include additional information in log files. You must be root or the LSF administrator to use this command.

-s *log_queue_size*

Specifies the maximum number of entries in the logging queue that the **mbatchd** logging thread uses. Specify an integer 100 - 500000. This value temporarily overrides the value of the **LSF_LOG_QUEUE_SIZE** parameter in the `lsf.conf` file. The logging queue contains the messages to be written to the log files.

If the **LSF_LOG_THREAD=N** parameter is defined in the `lsf.conf` file, the `-s` option is ignored.

See the **sbddebug** subcommand for an explanation of the other options.

For the -c option, the **mdbdebug** subcommand has the following valid log classes in addition to the valid log classes for the **sbddebug** subcommand:

LC2_EST
Log messages for the simulation-based estimator. You cannot use the **mbddebug** subcommand to change this log class.

LC2_G_FAIR
Log messages for global fairshare.

mbdhist [-t *time0*,*time1*] [-f *logfile_name*]
Displays historical events for the **mbatchd** daemon. Events describe the starting and exiting of the **mbatchd** daemon.

-t *time0*,*time1*

Displays only those events that occurred during the period from *time0* to *time1*. See the **bhist** command for the time format. The default is to display all queue events in the event log file.

-f *logfile_name*
Specify the file name of the event log file. Specify either an absolute or a relative path name. The default is to use the current event log file that is in the LSF system: LSB_SHAREDIR/*cluster_name*/logdir/lsb.events. Option -f is useful for offline analysis.

If you specified an administrator comment with the -C option of the **mbdrestart** subcommand, the **mbdhist** subcommand displays the comment text.

mbdrestart [-C *comment*] [-v] [-f] [-p | -s]
Dynamically reconfigures LSF and restarts the **mbatchd** and **mbschd** daemons. When live configuration with the **bconf** command is enabled (the **LSF_LIVE_CONFDIR** parameter is defined in the lsf.conffile), the **badmin mbdrestart** command uses the configuration files that are generated by the **bconf** command.

Configuration files are checked for errors and the results are printed to stderr. If no errors are found, configuration files are reloaded, the **mbatchd** and **mbschd** daemons are restarted, and events in the lsb.events file are replayed to recover the running state of the last **mbatchd** daemon. While the **mbatchd** daemon restarts, it is unavailable to service requests.

If warning errors are found, the **badmin** command prompts you to display detailed messages. If unrecoverable errors are found, the **mbatchd** and **mbschd** daemons do not restart, and the **badmin** command exits.

Important: If the lsb.events file is large, or many jobs are running, restarting the **mbatchd** daemon can take several minutes. If you need to reload only the configuration files, use the **badmin reconfig** command.

-C comment
Logs the text of comment as an administrator comment record to the lsb.events file. The maximum length of the comment string is 512 characters.

-v
Verbose mode. Displays detailed messages about the status of configuration files. All messages from configuration checking are printed to stderr.

-f
Disables interaction and forces reconfiguration and **mbatchd** daemon restart to proceed if configuration files contain no unrecoverable errors.

-p
Allows parallel **mbatchd** daemon restarts. Restart forks a child **mbatchd** daemon process to help minimize downtime for LSF. LSF starts a new or child **mbatchd** daemon process to read the configuration files and replay the event file. The old master **mbatchd** daemon can respond to client commands, handle job scheduling and status updates, dispatching, and updating new events to event files. When restart is complete, the child takes over as master **mbatchd** daemon, and the old master **mbatchd** daemon dies.

This option is the default behavior for **mbatchd** daemon restarts. Use the `-s` option to use serial **mbatchd** daemon restarts.

-s
Allows serial **mbatchd** daemon restarts. Use this option to change the default **mbatchd** deamon behavior, which is to restart in parallel.

mbdtime [-l *timing_level*] [-f *logfile_name*] [-o]
Sets timing level for the **mbatchd** daemon to include extra timing information in log files. You must be root or the LSF administrator to use this command.

perflog [-t sample_period] [-f logfile_name] [-d duration] | [-o]]
This feature is useful for troubleshooting large clusters where a cluster might not be responding due to **mbatchd** daemon performance problems. In such cases, the **mbatchd** daemon performance might be slow in handling high volume request, such as job submission, job status requests, and job `rusage` requests.

-t
Specifies the sampling period in minutes for performance metric collection. The default value is 5 minutes.

-f
Specifies a log file in which to save the information. It is either a file name or a full path file name. If you do not specify the path for the log file, then its default path is used. The default name for the log file is `mbatchd.perflog.<`*`host_name`*`>`.

The owner of the log file is the user who is specified in the **LSF_ADMIN** parameter. The log file permissions are the same as **mbatchd** daemon log permissions. Everyone has read and execute access, but the **LSF_ADMIN** user has write, read and execute access.

-d
The duration in minutes to keep logging performance metric data. The **mbatchd** daemon does not log messages after the duration expires, or until you stop it manually, restart the **mbatchd** daemon, or reconfigure with the **reconfig mbatchd** command. The default value for the duration is infinite. By default, performance metric data is always logged).

-o
Turns off dynamic performance metric logging (stop logging). If the **LSB_ENABLE_PERF_METRICS_LOG** parameter is enabled, logging returns to the static configuration.

perfmon start [*sample_period*] | setperiod *sample_period* | stop | view [-json]
Dynamically enables and controls scheduler performance metric collection.

Collecting and recording performance metric data might affect the performance of LSF. Smaller sampling periods can cause the `lsb.streams` file to grow faster.

The following metrics are collected and recorded in each sample period:

- The number of queries that are handled by **mbatchd**
- The number of queries for each of jobs, queues, and hosts. (**bjobs**, **bqueues**, and **bhosts** commands, and other daemon requests)
- The number of jobs submitted (divided into job submission requests and jobs submitted)
- The number of jobs dispatched
- The number of jobs reordered; that is, the number of jobs that reused the resource allocation of a finished job (the **RELAX_JOB_DISPATCH_ORDER** parameter in the `lsb.params` or `lsb.queues` file)
- The number of jobs completed
- The numbers of jobs that are sent to remote cluster
- The numbers of jobs that are accepted by from cluster
- The file descriptors that are used by the **mbatchd** daemon

- The following scheduler performance metrics are collected:
  - A shorter scheduling interval means that the job is processed more quickly
  - Number of different resource requirement patterns for jobs in use, which might lead to different candidate host groups. The more matching hosts that are required, the longer it takes to find them, which means a longer scheduling session.
  - Number of buckets (groups) in which jobs are put based on resource requirements and different scheduling policies. More buckets means a longer scheduling session.

start [*sample_period*]
Start performance metric collection dynamically and specify an optional sampling period in seconds for performance metric collection.

If no sampling period is specified, the default period set in the **SCHED_METRIC_SAMPLE_PERIOD** parameter in the `lsb.params` file is used.

stop
Stop performance metric collection dynamically.

view
Display performance metric information for the current sampling period.

When used with the `[-json]` option it will display the output in json format. This allows for better parsing and translation into graph format.

setperiod *sample_period*
Set a new sampling period in seconds.

qact [-C *comment*] [*queue_name* ... | all]
Activates a deactivated queue so that submitted jobs are dispatched from the queue. If the reserved word `all` is specified, the **qact** subcommand activates all queues. If no queue name is specified, the system `default` queue is activated. Jobs in a queue can be dispatched only if the queue is activated.

A queue that is inactivated by its run windows cannot be reactivated by this command.

-C *comment*
Logs the text of the comment as an administrator comment record to the `lsb.events` file. The maximum length of the comment string is 512 characters.

qclose [-C *comment*] [*queue_name* ... | all]
Closes a queue to prevent jobs from being submitted to the queue. If the reserved word `all` is specified, the **qclose** subcommand closes all queues. If no queue name is specified, the system `default` queue is closed. A queue does not accept submitted LSF jobs if it is closed.

-C *comment*
Logs the text as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

qhist [-t *time0*,*time1*] [-f *logfile_name*] [*queue_name* ...]
Displays historical events for specified queues, or for all queues if no queue is specified. Queue events are queue opening, closing, activating, and inactivating.

-t *time0*,*time1*

Displays only those events that occurred during the period from *time0* to *time1*. See the **bhist** command for the time format. The default is to display all queue events in the event log file.

-f *logfile_name*

Specifies the file name of the event log file. Either an absolute or a relative path name can be specified. The default is to use the current event log file in the LSF system: `LSB_SHAREDIR/`*cluster_name*`/logdir/lsb.events`. Option `-f` is useful for offline analysis.

If you specified an administrator comment with the `-C` option of the queue control subcommands **qclose**, **qopen**, **qact**, and **qinact**, the **qhist** subcommand displays the comment text.

qinact [-C *comment*] [*queue_name* ... | all]
Deactivates a queue to stop submitted jobs from being dispatched from the queue. If the reserved word `all` is specified, all queues are deactivated. If no queue name is specified, the system `default` queue is deactivated. Jobs in a queue cannot be dispatched if the queue is inactivated.

-C *comment*
Logs the text as an administrator comment record to the `lsb.events` file. The maximum length of the comment string is 512 characters.

qopen [-C *comment*] [*queue_name* ... | all]
Opens a closed queue so users can submit jobs to it. If the reserved word `all` is specified, the **qopen** subcommand opens all queues. If no queue name is specified, the system `default` queue is opened. A queue accepts submitted LSF jobs only if it is open.

-C *comment*
Logs the text of comment as an administrator comment record to `lsb.events`. The maximum length of the comment string is 512 characters.

quit
Exits the **badmin** command session.

rc error [-t *<days>*d | *<hours>*h | *<minutes>*m] [-p "*provider* ..."]
Shows LSF resource connector error messages from the host providers. These errors are provided by the third-party **mosquitto** message queue application, which must be running on the host.

-t *<days>*d | *<hours>*h | *<minutes>*m
Specifies the earliest time from which to retrieve the error messages.

Note: When specifying days, badmin retrieves messages from this time at midnight. For example, when running `badmin rc error -t 1d`, **badmin** retrieves messages from today at midnight, and when running `badmin rc error -t 2d`, **badmin** retrieves messages from yesterday at midnight.

-p "*provider* ...
Specifies the host providers from which to retrieve the error messages. Use a space to separate multiple host providers.

rc view [-c "instances | policies | templates ..."] [-p "*provider* ..."]
Shows LSF resource connector information from the host providers.

-c "instances | policies | templates ..."
Specifies whether to view information on instances, policies, or templates. Use a space to separate multiple types of information. By default, this command shows information on instances only. If `policies` is selected with the `-c` option, the `-p` option is ignored because all policies are displayed, not just for the specified providers.

-p "*provider* ..."
Specifies the host providers from which to view information. Use a space to separate multiple host providers. If `policies` is selected with the `-c` option, the `-p` option is ignored because all policies are displayed, not just for the specified providers.

reconfig [-v] [-f]
Dynamically reconfigures LSF.

Configuration files are checked for errors and the results are displayed to `stderr`. If no errors are found in the configuration files, a reconfiguration request is sent to the **mbatchd** daemon and configuration files are reloaded. When live configuration with the **bconf** command is enabled (the **LSF_LIVE_CONFDIR** parameter is defined in the `lsf.conf`file), the **badmin reconfig** command uses the configuration files that are generated by the **bconf** command.

Important: The **reconfig** subcommand does not restart the **mbatchd** daemon and does not replay the `lsb.events` file. To restart the **mbatchd** daemon and replay the `lsb.events` file, use the **badmin mbdrestart** command.

When you use this command, the **mbatchd** daemon is available to service requests while reconfiguration files are reloaded. Configuration changes made since system boot or the last reconfiguration take effect.

If warning errors are found, the **badmin** command prompts you to display detailed messages. If unrecoverable errors are found, reconfiguration fails, and the **badmin** command exits.

If you add a host to a queue or to a host group or compute unit, the new host is not recognized by jobs that were submitted before you reconfigured. If you want the new host to be recognized, you must use the command **badmin mbdrestart**.

Resource requirements that are determined by the queue no longer apply to a running job after you use the **badmin reconfig** command. For example, if you change the **RES_REQ** parameter in a queue and reconfigure the cluster, the previous queue-level resource requirements for running jobs are lost.

-v
Verbose mode. Displays detailed messages about the status of the configuration files. Without this option, the default is to display the results of configuration file checking. All messages from the configuration file check are printed to `stderr`.

-f
Disables interaction and proceeds with reconfiguration if configuration files contain no unrecoverable errors.

sbddebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o] [*host_name* ...]
Sets the message log level for the **sbatchd** daemon to include additional information in log files. You must be `root` or the LSF administrator to use this command.

In LSF multicluster capability, debug levels can be set only for hosts within the same cluster. For example, you cannot set debug or timing levels from a host in `clusterA` for a host in `clusterB`. You need to be on a host in `clusterB` to set up debug or timing levels for `clusterB` hosts.

If the command is used without any options, the following default values are used:

***class_name*=0**
No additional classes are logged.

***debug_level*=0**
LOG_DEBUG level in parameter **LSF_LOG_MASK**.

logfile_name*daemon_name.log.*host_name
LSF system log file in the LSF system log file directory, in the format *daemon_name*.log.*host_name*.

host_name*=*local_host
Host from which the command was submitted.

-c *class_name* ...
Specify software classes for which debug messages are to be logged.

Note: Classes are also listed in the `lsf.h` header file.

By default, no additional classes are logged (class name `0`).

The following log classes are supported:

LC_ADVRSV and LC2_ADVRSV
Log advance reservation modifications.

LC2_AFFINITY
Log messages that are related to affinity.

LC_AFS and LC2_AFS
Log AFS messages.

LC_AUTH and LC2_AUTH
Log authentication messages.

LC_CHKPNT and LC2_CHKPNT
Log checkpointing messages.

LC_COMM and LC2_COMM
Log communication messages.

LC_DCE and LC2_DCE
Log messages that pertain to DCE support.

LC_EEVENTD and LC2_EEVENTD
Log **eeventd** daemon messages.

LC_ELIM and LC2_ELIM
Log ELIM messages.

LC_EXEC and LC2_EXEC
Log significant steps for job execution.

LC_FAIR
Log fairshare policy messages.

LC_FILE and LC2_FILE
Log file transfer messages.

LC2_GUARANTEE
Log messages that are related to guaranteed SLAs.

LC_HANG and LC2_HANG
Mark where a program might hang.

LC_JARRAY and LC2_JARRAY
Log job array messages.

LC_JLIMIT and LC2_JLIMIT
Log job slot limit messages.

LC_LOADINDX and LC2_LOADINDX
Log load index messages.

LC_M_LOG and LC2_M_LOG
Log multievent log messages.

LC_MEMORY and LC2_MEMORY
Log messages that are related to MEMORY allocation.

LC_MPI and LC2_MPI
Log MPI messages.

LC_MULTI and LC2_MULTI
Log messages that pertain to LSF multicluster capability.

LC_PEND and LC2_PEND
Log messages that are related to job pending reasons.

LC_PERFM and LC2_PERFM
Log performance messages.

LC_PIM and LC2_PIM
Log PIM messages.

LC_PREEMPT and LC2_PREEMPT
Log preemption policy messages.

LC2_RC
Log resource connector messages.

LC_RESOURCE and LC2_RESOURCE
Log messages that are related to resource broker.

LC_RESREQ and LC2_RESREQ
Log resource requirement messages.

LC_SCHED and LC2_SCHED
Log messages that pertain to the batch scheduler.

LC_SIGNAL and LC2_SIGNAL
Log messages that pertain to signals.

LC_SYS and LC2_SYS
Log system call messages.

LC_TRACE and LC2_TRACE
Log significant program walk steps.

LC_XDR and LC2_XDR
Log everything that is transferred by XDR.

LC_XDRVERSION and LC2_XDRVERSION
Log messages for XDR version.

-l ***debug_level***
Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower levels.

The default is `0` (`LOG_DEBUG` level in parameter **LSF_LOG_MASK**)

The following values are supported:

0
`LOG_DEBUG` level for parameter **LSF_LOG_MASK** in the `lsf.conf` file. `0` is the default.

1
`LOG_DEBUG1` level for extended logging. A higher level includes lower logging levels. For example, `LOG_DEBUG1` includes `LOG_DEBUG` levels.

2
`LOG_DEBUG2` level for extended logging. A higher level includes lower logging levels. For example, `LOG_DEBUG2` includes `LOG_DEBUG1` and `LOG_DEBUG` levels.

3
`LOG_DEBUG3` level for extended logging. A higher level includes lower logging levels. For example, `LOG_DEBUG3` includes `LOG_DEBUG2`, `LOG_DEBUG1`, and `LOG_DEBUG` levels.

-f ***logfile_name***
Specify the name of the file into which debugging messages are to be logged. A file name with or without a full path might be specified.

If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file that is created has the following format:

```
logfile_name.daemon_name.log.host_name
```

On UNIX and Linux, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

By default, current LSF system log file in the LSF system log file directory is used.

-o
Turns off temporary debug settings and resets them to the daemon start state. The message log level is reset back to the value of **LSF_LOG_MASK** and classes are reset to the value of **LSB_DEBUG_MBD**, **LSB_DEBUG_SBD**.

The log file is also reset back to the default log file.

host_name ...
Optional. Sets debug settings on the specified host or hosts.

The default is the local host (the host from which command was submitted).

sbdtime [-l *timing_level*] [-f *logfile_name*] [-o] [*host_name* ...]

Sets the timing level for the **sbatchd** daemon to include extra timing information in log files. You must be `root` or the LSF administrator to use this command.

In LSF multicluster capability, timing levels can be set only for hosts within the same cluster. For example, you cannot set debug or timing levels from a host in `clusterA` for a host in `clusterB`. You need to be on a host in `clusterB` to set up debug or timing levels for `clusterB` hosts.

If the command is used without any options, the following default values are used:

***timing_level*=no**
Timing information is recorded.

***logfile_name*=current**
LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*.

***host_name*=local**
The host from which command was submitted.

-l *timing_level*
Specifies detail of timing information that is included in log files. Timing messages indicate the execution time of functions in the software and are logged in milliseconds.

The following values are supported: 1 | 2 | 3 | 4 | 5

The higher the number, the more functions in the software that are timed and whose execution time is logged. The lower numbers include more common software functions. Higher levels include all lower levels.

By default, no timing information is logged.

-f *logfile_name*
Specify the name of the file into which timing messages are to be logged. A file name with or without a full path can be specified.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file that is created has the following format:

```
logfile_name.daemon_name.log.host_name
```

On UNIX and Linux, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

Note: Both timing and debug messages are logged in the same files.

The default is the current LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*.

-o
Optional. Turn off temporary timing settings and reset them to the daemon start state. The timing level is reset back to the value of the parameter for the corresponding daemon (**LSB_TIME_MBD**, **LSB_TIME_SBD**).

The log file is also reset back to the default log file.

***host_name* ...**
Sets the timing level on the specified host or hosts.

The default is the local host from which command was submitted).

schddebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o] [-s *log_queue_size*]

Sets message log level for the **mbschd** demon to include additional information in log files. You must be `root` or the LSF administrator to use this command.

-s *log_queue_size*
Specifies the maximum number of entries in the logging queue that is used by the **mbschd** daemon logging thread. Specify an integer 100 - 500000. The logging queue contains the messages to be written to the log files.

This option is ignored if the LSF_LOG_THREAD=N parameter is defined in the `lsf.conf` file.

See the **sbddebug** subcommand for an explanation of the other options.

schdtime [-l *timing_level*] [-f] [-o]
Sets timing level for the **mbschd** daemon to include extra timing information in log files. You must be root or the LSF administrator to use this command.

See the description of the **sbdtime** subcommand for an explanation of options.

showconf mbd | [sbd [*host_name* ... | all] | gpd]
Display all configured parameters and their values set in the `lsf.conf` or `ego.conf` file that affect the **mbatchd**, **sbatchd**, and **gpolicyd** daemons.

In LSF multicluster capability, the **badmin showconf** command displays only the parameters of daemons on the local cluster.

Running the **badmin showconf** command from a master candidate host reaches all server hosts in the cluster. Running the **badmin showconf** command from a slave-only host might not be able to reach other slave-only hosts.

The **badmin showconf** command displays only the values that are used by LSF.

The **badmin showconf** command displays the value of the **EGO_MASTER_LIST** parameter from wherever it is defined. You can define either the **LSF_MASTER_LIST** parameter or the **EGO_MASTER_LIST** parameter in the `lsf.conf` file. If EGO is enabled in the LSF cluster, LIM reads the `lsf.conf` file first, then the `ego.conf` file. The value of the **LSF_MASTER_LIST** parameter is displayed only if the **EGO_MASTER_LIST** parameter is not defined at all in the `ego.conf` file.

For example, if you define the **LSF_MASTER_LIST** parameter in the `lsf.conf`file, and the **EGO_MASTER_LIST** parameter in the `ego.conf` file, the **badmin showconf** command displays the value of the **EGO_MASTER_LIST** parameter.

If EGO is enabled in the LSF cluster, and you define the **LSF_MASTER_LIST** parameter in the `lsf.conf` file, and the **EGO_MASTER_LIST** parameter in the `ego.conf` file, the **badmin showconf** command displays the value of the **EGO_MASTER_LIST** parameter in the `ego.conf` file.

If EGO is disabled, the `ego.conf` file is not loaded, so parameters that are defined in the `lsf.conf` file are displayed.

showstatus
Displays current LSF runtime information about the whole cluster, including information about hosts, jobs, users, user groups, simulation-based estimation, and **mbatchd** daemon startup and reconfiguration.

See also

bhosts, **bqueues**, `lsb.hosts`, `lsb.params`, `lsb.queues`, `lsf.cluster`, `lsf.conf`, **sbatchd**, **mbatchd**, **mbschd**

Chapter 3. bapp

Displays information about application profile configuration.

Synopsis

bapp [-alloc] [-l | -w] [*application_profile* ...]

bapp [-h | -V]

Description

Displays information about application profiles that are configured in the `lsb.applications` file.

Returns application name, task statistics, and job state statistics for all application profiles.

In the LSF multicluster capability, returns the information about all application profiles in the local cluster.

Returns job slot statistics if the `-alloc` option is used.

CPU time is normalized.

Options

-alloc

Shows counters for slots in RUN, SSUSP, USUSP, and RSV state. The slot allocation is different depending on whether the job is an exclusive job or not.

-w

Wide format. Fields are displayed without truncation.

-l

Long format with additional information.

Displays the following additional information about the application profile:

- Description
- Profile characteristics and statistics
- Parameters
- Resource usage limits
- Associated commands
- Binding policy
- **NICE** value
- Job controls
- Pending time limits
- Eligible pending time limits
- APS priority factors

application_profile ...

Displays information about the specified application profile.

-h

Prints command usage to `stderr` and exits.

-V

Prints product release version to `stderr` and exits.

Default output format

Displays the following fields:

APPLICATION_NAME

The name of the application profile. Application profiles are named to correspond to the type of application that usually runs within them.

NJOBS

The total number of tasks for all jobs that are currently held in the application profile. The total includes tasks in pending, running, and suspended jobs.

If the `-alloc` option is used, total is the sum of the counters for RUN, SSUSP, USUSP, and RSV states.

PEND

The number of tasks for all pending jobs in the application profile. If used with the `-alloc` option, output is `0`, because pending jobs do not have slot allocation.

RUN

The number of tasks for all running jobs in the application profile. If the `-alloc` option is used, the total is allocated slots for the jobs in the application profile.

SUSP

The number of tasks for all suspended jobs in the application profile. If the `-alloc` option is used, the total is allocated slots for the jobs in the application profile.

Long output format with the -l option

In addition to the default output fields, the `-l` option displays the following fields:

Description

A description of the typical use of the application profile.

PARAMETERS/STATISTICS

SSUSP

The number of tasks for all jobs in the application profile that are suspended by LSF because of load levels or run windows.

USUSP

The number of tasks for all jobs in the application profile that are suspended by the job submitter or by the LSF administrator.

RSV

The number of tasks that reserve slots for pending jobs in the application profile.

ENV_VARS

The name-value pairs that are defined by the application-specific environment variables.

Per-job resource usage limits

The soft resource usage limits that are imposed on the jobs that are associated with the application profile. These limits are imposed on a per-job and a per-process basis.

The following per-job limits are supported:

CPULIMIT

The maximum CPU time a job can use, in minutes, relative to the CPU factor of the named host. The `CPULIMIT` is scaled by the CPU factor of the execution host so that jobs are allowed more time on slower hosts.

MEMLIMIT

The maximum running set size (RSS) of a process.

By default, the limit is shown in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for display (MB, GB, TB, PB, or EB).

MEMLIMIT_TYPE

A memory limit is the maximum amount of memory a job is allowed to consume. Jobs that exceed the level are killed. You can specify different types of memory limits to enforce, based on PROCESS, TASK, or JOB (or any combination of the three).

PROCESSLIMIT

The maximum number of concurrent processes that are allocated to a job.

SWAPLIMIT

The swap space limit that a job can use.

By default, the limit is shown in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for display (MB, GB, TB, PB, or EB).

TASKLIMIT

The maximum number of tasks that are allocated to a job.

THREADLIMIT

The maximum number of concurrent threads that are allocated to a job.

Per-process resource usage limits

The following UNIX per-process resource limits are supported:

CORELIMIT

The maximum size of a core file.

By default, the limit is shown in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for display (MB, GB, TB, PB, or EB).

DATALIMIT

The maximum size of the data segment of a process, in KB. This limit restricts the amount of memory a process can allocate.

FILELIMIT

The maximum file size a process can create, in KB.

RUNLIMIT

The maximum wall clock time a process can use, in minutes. RUNLIMIT is scaled by the CPU factor of the execution host.

STACKLIMIT

The maximum size of the stack segment of a process. This limit restricts the amount of memory a process can use for local variables or recursive function calls.

By default, the limit is shown in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for display (MB, GB, TB, PB, or EB).

BIND_JOB

The processor binding policy for sequential and parallel job processes enabled in the application profile. Displays one of the following values: NONE, BALANCE, PACK, ANY, USER, or USER_CPU_LIST.

```
bapp -l app1
APPLICATION NAME: app1
 -- test processor binding options
```

```
...
PARAMETERS:
BIND_JOB: ANY
```

For compatibility with an earlier versions, the **bapp -l** command displays Y or N if the **BIND_JOB** parameter is defined with those values in the application profile.

CHKPNT_DIR

The checkpoint directory, if automatic checkpointing is enabled for the application profile.

CHKPNT_INITPERIOD

The initial checkpoint period in minutes. The periodic checkpoint does not happen until the initial period elapses.

CHKPNT_PERIOD

The checkpoint period in minutes. The running job is checkpointed automatically every checkpoint period.

CHKPNT_METHOD

The checkpoint method.

MIG

The migration threshold in minutes. A value of 0 (zero) specifies that a suspended job is to be migrated immediately.

Where a host migration threshold is also specified, and is lower than the job value, the host value is used.

PRE_EXEC

The job-based pre-execution command for the application profile. The **PRE_EXEC** command runs on the execution host before the job that is associated with the application profile is dispatched to the execution host (or to the first host selected for a parallel batch job).

POST_EXEC

The job-based post-execution command for the application profile. The **POST_EXEC** command runs on the execution host after the job finishes.

HOST_PRE_EXEC

The host-based pre-execution command for the application profile. The **HOST_PRE_EXEC** command runs on all execution hosts before the job that is associated with the application profile is dispatched to the execution hosts. If a job-based pre-execution **PRE_EXEC** command was defined at the queue-level/application-level/job-level, the **HOST_PRE_EXEC** command runs before the **PRE_EXEC** command of any level. The host-based pre-execution command cannot be run on Windows systems.

HOST_POST_EXEC

The host-based post-execution command for the application profile. The **HOST_POST_EXEC** command runs on the execution hosts after the job finishes. If a job-based post-execution **POST_EXEC** command was defined at the queue-level/application-level/job-level, the **HOST_POST_EXEC** command runs after the **POST_EXEC** command of any level. The host-based post-execution command cannot be run on Windows systems.

LOCAL_MAX_PREEXEC_RETRY_ACTION

The action to take on a job when the number of times to attempt its pre-execution command on the local cluster (the value of the **LOCAL_MAX_PREEXEC_RETRY** parameter) is reached.

JOB_INCLUDE_POSTPROC

If the **JOB_INCLUDE_POSTPROC=Y** parameter is defined, post-execution processing of the job is included as part of the job.

JOB_POSTPROC_TIMEOUT

Timeout in minutes for job post-execution processing. If post-execution processing takes longer than the timeout, the **sbatchd** daemon reports that post-execution failed (POST_ERR status). On UNIX, it kills the process group of the job's post-execution processes. On Windows, only the parent process of the pre-execution command is killed when the timeout expires, the child processes of the pre-execution command are not killed.

REQUEUE_EXIT_VALUES

Jobs that exit with these values are automatically requeued.

RES_REQ

Resource requirements of the application profile. Only the hosts that satisfy these resource requirements can be used by the application profile.

JOB_STARTER

An executable file that runs immediately before the batch job, taking the batch job file as an input argument. All jobs that are submitted to the application profile are run through the job starter, which is used to create a specific execution environment before LSF processes the jobs themselves.

CHUNK_JOB_SIZE

Chunk jobs only. Specifies the maximum number of jobs that are allowed to be dispatched together in a chunk job. All of the jobs in the chunk are scheduled and dispatched as a unit rather than individually.

RERUNNABLE

If the RERUNNABLE field displays yes, jobs in the application profile are automatically restarted or rerun if the execution host becomes unavailable. However, a job in the application profile is not restarted if you use **bmod** to remove the rerunnable option from the job.

RESUME_CONTROL

The configured actions for the resume job control.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is RESUME.

SUSPEND_CONTROL

The configured actions for the suspend job control.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is SUSPEND.

TERMINATE_CONTROL

The configured actions for terminate job control.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is TERMINATE.

NO_PREEMPT_INTERVAL

The configured uninterrupted running time (minutes) that must pass before a preemptable job can be preempted.

MAX_TOTAL_TIME_PREEMPT

The configured maximum total preemption time (minutes) after which preemption cannot take place.

NICE

The relative scheduling priority at which jobs from the application run.

Current working directory (CWD) information

JOB_CWD

The current working directory for the job in the application profile. The path can be absolute or relative to the submission directory, and includes dynamic patterns.

JOB_CWD_TTL

The time to live for the current working directory for a job. LSF cleans the created CWD after a job finishes based on the TTL value.

JOB_SIZE_LIST

A list of job sizes (number of tasks) allowed on this application, including the default job size that is assigned if the job submission does not request a job size. Configured in the `lsb.applications` file.

PEND_TIME_LIMIT

The pending time limit for a job in the application profile. If a job remains pending for longer than this specified time limit, IBM Spectrum LSF RTM (LSF RTM) triggers the alarm and other actions. Configured in the `lsb.applications` file.

ELIGIBLE_PEND_TIME_LIMIT

The eligible pending time limit for a job in the application profile. If a job remains in an eligible pending state for longer than this specified time limit, IBM Spectrum LSF RTM (LSF RTM) triggers the alarm and other actions. Configured in the `lsb.applications` file.

PRIORITY

The APS priority factor for the application profile. PRIORITY is defined by the **PRIORITY** parameter in the configuration file `lsb.applications`.

See also

`lsb.applications`, `lsb.queues`, **bsub**, **bjobs**, **badmin**, **mbatchd**

Chapter 4. bbot

Moves a pending job to the bottom of the queue relative to the last job in the queue.

Synopsis

`bbot [-u] job_ID | "job_ID[index_list]" [position]`

`bbot -h | -V`

Description

Changes the queue position of a pending job or job array element to affect the order in which jobs are considered for dispatch.

By default, LSF dispatches jobs in a queue in the order of arrival (that is, first-come, first-served), subject to availability of suitable server hosts.

The **bbot** command allows users and the LSF administrator to manually change the order in which jobs are considered for dispatch. Users can operate only on their own jobs. LSF administrators can operate on any jobs.

If used by the LSF administrator, the **bbot** command moves the selected job after the last job with the same priority in the queue.

If used by a user, the **bbot** command moves the selected job after the last job submitted by the user with the same priority in the queue.

Pending jobs are displayed by the **bjobs** command in the order that they are considered for dispatch.

You can use the **bbot** command to change the dispatch order of fairshare jobs. However, if a fairshare job is moved by the LSF administrator with the **btop**command, the job is not subject to fairshare scheduling unless the same job is later moved by the LSF administrator with the **bbot**command. In this case, the job is scheduled again with the same fairshare policy.

To prevent users from changing the queue position of a pending job with the **bbot** command, configure the **JOB_POSITION_CONTROL_BY_ADMIN=Y** parameter in the `lsb.params` file.

You cannot run the **bbot** command on jobs that are pending in an absolute priority scheduling (APS) queue.

Options

-u

Optional. When specified, allows jobs to be moved relative to the normal user's own job list. This option can only be used by the LSF administrator. If used by a normal user, this option is ignored.

***job_ID* | "*job_ID*[*index_list*]"**

Required. Job ID of the job or job array on which to operate.

For a job array, the index list, the square brackets, and the quotation marks are required. An index list is used to operate on a job array. The index list is a comma-separated list whose elements have the following syntax:

```
start_index[-end_index[:step]]
```

The *start_index*, *end_index*, and *step* values are positive integers. If the step is omitted, a step of one is assumed. The job array index starts at one. The maximum job array index is 1000. All jobs in the array have the same job ID and parameters. Each element of the array is distinguished by its array index.

position

Optional. The position argument can be specified to indicate where in the queue the job is to be placed. The value of *position* is a positive number that indicates the target position of the job from the end of the queue. The positions are relative to only the applicable jobs in the queue, depending on whether the invoker is a regular user or the LSF administrator. The default value of 1 means the position is after all other jobs with the same priority.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

bjobs, **bswitch**, **btop**, **JOB_POSITION_CONTROL_BY_ADMIN** parameter in the `lsb.params` file

Chapter 5. bchkpnt

Checkpoints one or more checkpointable jobs

Synopsis

`bchkpnt` [`-f`] [`-k`] [`-app` *application_profile_name*] [`-p` *minutes* | `-p 0`] *job_ID* | "*job_ID*[*index_list*]" ...

`bchkpnt` [`-f`] [`-k`] [`-app` *application_profile_name*] [`-p` *minutes* | `-p 0`] `-J` *job_name* |`-m` *host_name* | `-m` *host_group* |`-q` *queue_name* |`-u` "*user_name*" | `-u all` [`0`]

`bchkpnt -h` | `-V`

Description

Checkpoints the most recently submitted running or suspended checkpointable job.

LSF administrators and **root** can checkpoint jobs that are submitted by other users.

Jobs continue to run after they are checkpointed.

LSF runs the **echkpnt** executable file that is found in the **LSF_SERVERDIR** directory to checkpoint the job.

Only running members of a chunk job can be checkpointed. For chunk jobs in `WAIT` state, the **mbatchd** daemon rejects the checkpoint request.

Options

0

(Zero). Checkpoints all of the jobs that satisfy other specified criteria.

-f

Forces a job to be checkpointed even if non-checkpointable conditions exist (these conditions are operating system-specific).

-app *application_profile_name*

Operates only on jobs that are associated with the specified application profile. You must specify an existing application profile. If *job_ID* or `0` is not specified, only the most recently submitted qualifying job is operated on.

-k

Kills a job after it is successfully checkpointed.

-p *minutes* | -p 0

Enables periodic checkpointing and specifies the checkpoint period, or modifies the checkpoint period of a checkpointed job. Specify `-p 0` (zero) to disable periodic checkpointing.

Checkpointing is a resource-intensive operation. For your job to make progress while still providing fault tolerance, specify a checkpoint period of 30 minutes or longer.

-J *job_name*

Checkpoints only jobs that have the specified job name.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (`*`) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-m *host_name* | -m *host_group*

Checkpoints only jobs that are dispatched to the specified hosts.

-q *queue_name*

Checkpoints only jobs that are dispatched from the specified queue.

-u "*user_name*" | -u all

Checkpoints only jobs that are submitted by the specified users. The keyword `all` specifies all users. Ignored if a job ID other than 0 (zero) is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

***job_ID* | "*job_ID*[*index_list*]"**

Checkpoints only the specified jobs.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

```
bchkpnt 1234
```

Checkpoints the job with job ID 1234.

```
bchkpnt -p 120 1234
```

Enables periodic checkpointing or changes the checkpoint period to 120 minutes (2 hours) for a job with job ID 1234.

```
bchkpnt -m hostA -k -u all 0
```

When used by root or the LSF administrator, checkpoints and kills all checkpointable jobs on `hostA`. This command is useful when a host needs to be shut down or rebooted.

See also

bsub, **bmod**, **brestart**, **bjobs**, **bqueues**, bhosts(1), `lsb.queues`, **mbatchd**

Chapter 6. bclusters

Displays information about IBM Spectrum LSF multicluster capability

Synopsis

`bclusters` [`-app`] [`-w`]

`bclusters` [`-h` | `-V`]

Description

For the job forwarding model, displays a list of LSF multicluster capability queues together with their relationship with queues in remote clusters.

For the resource leasing model, displays remote resource provider and consumer information, resource flow information, and connection status between the local and remote cluster.

Options

-app

Displays available application profiles in remote clusters.

-w

Wide format. Displays the information in a wide format.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output: Job forwarding information

Displays a list of LSF multicluster capability queues together with their relationship with queues in remote clusters.

Information that is related to the job forwarding model is displayed under the heading Job Forwarding Information.

LOCAL_QUEUE

Name of a local LSF multicluster capability send-jobs or receive-jobs queue.

JOB_FLOW

Indicates direction of job flow.

send

The local queue is a LSF multicluster capability send-jobs queue (SNDJOBS_TO is defined in the local queue).

recv

The local queue is a LSF multicluster capability receive-jobs queue (RCVJOBS_FROM is defined in the local queue).

REMOTE

For send-jobs queues, shows the name of the receive-jobs queue in a remote cluster.

For receive-jobs queues, always a dash (-).

CLUSTER

For send-jobs queues, shows the name of the remote cluster that contains the receive-jobs queue.

For receive-jobs queues, shows the name of the remote cluster that can send jobs to the local queue.

STATUS

Indicates the connection status between the local queue and remote queue.

ok

The two clusters can exchange information and the system is properly configured.

disc

Communication between the two clusters is not established. The `disc` status might occur because no jobs are waiting to be dispatched, or because the remote master cannot be located.

reject

The remote queue rejects jobs from the send-jobs queue. The local queue and remote queue are connected and the clusters communicate, but the queue-level configuration is not correct. For example, the send-jobs queue in the submission cluster points to a receive-jobs queue that does not exist in the remote cluster.

If the job is rejected, it returns to the submission cluster.

Output: Resource lease information

Displays remote resource provider and consumer information, resource flow information, and connection status between the local and remote cluster.

Information that is related to the resource leasing model is displayed under the heading Resource Lease Information.

REMOTE_CLUSTER

For borrowed resources, name of the remote cluster that is the provider.

For exported resources, name of the remote cluster that is the consumer.

RESOURCE_FLOW

Indicates direction of resource flow.

IMPORT

Local cluster is the consumer and borrows resources from the remote cluster (**HOSTS** parameter in one or more local queue definitions includes remote resources).

EXPORT

Local cluster is the provider and exports resources to the remote cluster.

STATUS

Indicates the connection status between the local and remote cluster.

ok

LSF multicluster capability jobs can run.

disc

No communication between the two clusters. The `disc` status might be a temporary situation or might indicate a LSF multicluster capability configuration error.

conn

The two clusters communicate, but the lease is not established. The `conn` status is typically a temporary situation, lasting only until jobs are submitted.

Output: Remote cluster application information

The **bcluster -app** command displays information that is related to application profile configuration under the heading `Remote Cluster Application Information`. Application profile information is only displayed for the job forwarding model. The **bclusters** command does not show local cluster application profile information.

REMOTE_CLUSTER

The name of the remote cluster.

APP_NAME

The name of the application profile available in the remote cluster.

DESCRIPTION

The description of the application profile.

Files

Reads the `lsb.queues` and `lsb.applications` files.

See also

bapp, **bhosts**, **bqueues**, **lsclusters**, **lsinfo**, `lsb.applications`, `lsb.queues`

Chapter 7. bconf

Submits live reconfiguration requests, updating configuration settings in active memory without restarting daemons.

Synopsis

`bconf` *action object_type=object_name* "*value_pair*[`;`*value_pair*...]"] [`-c` "*comment*"] [`-f`]

`bconf hist` [`-l`|`-w`] [`-o` *object_type*] [`-u` *user_name*] [`-T` *time_period*] [`-a` *action*] [`-f` *config_file*] [*history_file*]

`bconf disable`

`bconf -h` [*action* [*object_type*]]

`bconf -pack` *pack_file_name*

`bconf -V`

Action synopsis

`addmember usergroup` | `hostgroup` | `queue` | `limit` | `gpool` | `serviceclass=`*object_name* "*value_pair*[`;`*value_pair* ...]" [`-c` "*comment*"]

`rmmember usergroup` | `hostgroup` | `queue` | `limit` | `gpool` | `serviceclass=`*object_name* "*value_pair*[`;`*value_pair* ...]" [`-c` "*comment*"]

`update user` | `usergroup` | `host` | `hostgroup` | `queue` | `limit` | `gpool` | `serviceclass=`*object_name* "*value_pair*[`;`*value_pair* ...]" [`-c` "*comment*"]

`create usergroup` | `limit` | `serviceclass=`*object_name* "*value_pair*[`;`*value_pair* ...]" [`-c` "*comment*"]

`delete usergroup` | `limit` | `serviceclass=`*object_name* "*value_pair*[`;`*value_pair* ...]" [`-c` "*comment*"] [`-f`]

`set user` |`usergroup` | `limit=`*object_name* "*value_pair*[`;`*value_pair* ...]" [`-c` "*comment*"]

`add host=`*object_name* "*value_pair*[`;`*value_pair* ...]" [`-c` "*comment*"]

Description

The **bconf** command is enabled when the **LSF_LIVE_CONFDIR** parameter is defined in the `lsf.conf` file.

The **bconf** command allows configuration changes without restarting LSF or any daemons. Changes are made in active LSF memory, and updated configuration files are written to the directory defined by parameter **LSF_LIVE_CONFDIR**. Original configuration files are not changed. However, LSF reloads files found in the **LSF_LIVE_CONFDIR** directory during restart or reconfiguration in place of permanent configuration files.

Important: Before you enable live reconfiguration, run the **badmin reconfig** command to make sure that all configuration files have no error or warning messages. Merge multiple sections in configuration files where possible. Keep the order of sections and the syntax used in the configuration file templates in all configuration files used with live reconfiguration.

Configuration changes made by using the **bconf** command cannot be rolled back. Use the **bconf** command to reverse the configuration requests and undo unwanted configuration changes. You can also manually remove or replace the configuration files in the **LSF_LIVE_CONFDIR** directory before restart or reconfiguration.

The first **bconf** command that is executed after restart or reconfiguration backs up the files that were loaded into memory. All files that the **bconf** command can change are backed up in the

LSF_LIVE_CONFDIR directory as `*.bak` files. The backup files always represent the configuration before **bconf** commands were run.

Cluster administrators can run all **bconf** commands. All users can run **bconf hist** queries. All **bconf** command requests must be made from static servers.

The `gpool` administrators can manage the corresponding guaranteed resource pool.

User group administrators with `usershares` rights can adjust user shares.

User group administrators with `full` rights can change the following configurations:

- Adjust user shares and group members
- Delete the user group
- Create new user groups

User group administrators with `full` rights can add a user group member to the user group only if they also have `full` rights for the member user group. User group administrators who add a user group through the **bconf create** command are automatically added to the **GROUP_ADMIN** list with `full` rights for the new user group.

Important: Remove the configuration files in the **LSF_LIVE_CONFDIR** directory, or merge files into the **LSF_CONFDIR** directory before you disable live configuration, upgrade LSF, apply patches to LSF, or add server hosts.

The **bconf** command supports common configuration changes. The **bconf** command cannot change all configurations. For time-based configuration, global configurations are changed globally, and configuration for the active time window are changed only for the time window.

The **bconf** command changes the following configuration files:

- `lsb.resources`
- `lsb.queues`
- `lsb.users`
- `lsb.hosts`
- `lsf.cluster.`*`clustername`*
- `lsb.serviceclasses`

Important: Changing these configuration files manually while live reconfiguration is enabled automatically disables the **bconf** command. Further live reconfiguration requests are rejected.

The **bconf** command changes *objects*, or configuration blocks enclosed in `Begin` and `End` statements, in the configuration files. One **bconf** request can affect several configured objects. For example, deleting a user group that appears in the configuration for a limit and a queue also changes the limit and queue configuration, and returns the following confirmation messages:

```
bconf delete usergroup=ug1
bconf: Request to delete usergroup <ug1> impacts the following:
    <USERS> in limit <limit1>
    <USERS FAIRSHARE > in queue <big_mem_queue>
Are you sure you want to delete usergroup <ug1> (y/n)?
```

The API corresponding to the **bconf** command is `lsb_liveconfig`..

Subcommands and options

***action object_type=object_name* "*value_pair*[;*value_pair*...]"] [-c "*comment*"] [-f]**

action

action is the requested action that is supported by live reconfiguration. The following keywords can be an *action* value: `addmember`, `rmmember`, `update`, `create`, `add`, `delete, set`.

addmember
Adds a member to the group or list of an existing field in an object, or updates the value of an existing member.

Cannot be used with reserved words such as `all`, excluded elements such as `~user1` or `! host1`, or members defined by regular expressions such as `hostA[01-10]` or `hostA*`.

When used with an existing member, the value of the member is updated within the object.

rmmember
Removes a member from the group or list of an existing key (field) in an object.

Restriction: You cannot remove all members from groups and lists cannot have all (except USER_SHARES). The group or list cannot contain only reserved words such as `others`, `all`, or `allremote`, or contain only excluded members.

You cannot use the `rmmember` option with reserved words such as `all`, excluded elements such as `~user1` or `!host1`, or members defined by regular expressions such as `hostA[01-10]` or `hostA*`. Hosts added with the **badmin hghostadd** command cannot be removed with the **bconf rmmember** command.

update
Updates by replacing the old value with the new value, or adding the field if it is not already configured.

Use the **update usergroup=***group_name* or **update hostgroup=***group_name* command to reload an external user group.

create
Creates a new object.

add
Adds a new host.

delete
Deletes an existing object.

You cannot delete a user group under the following conditions:

- The user group contains running or pending jobs (run the **busers** command to check)
- The user group appears in a LSF multicluster capability `UserMap` section in the `lsb.users` file
- The user group is the default user group defined by the **DEFAULT_USER_GROUP** parameter in the `lsb.params` file.

Deleted user groups are counted towards the maximum allowed number of user groups until the next **restart** or **reconfig** command is run. Deleted user groups might still show in the **busers** command output.

set
Forces an update or a create action.

object_type
Any block that is enclosed by `BeginSection ... EndSection` in a configuration file that is changed by a **bconf** command request. An object includes a type and name and contains attributes that are called *keys*, which are fields that are defined in the object section of the file. The following keywords can be an *object_type* value: `user`, `usergroup`, `host`, `hostgroup`, `queue`, `limit`, `gpool`. Not all actions apply to all object types.

user
Can be used with the following actions and keywords:

- *action* - `update`, `set`
- *value_pair* - the following keywords in the `lsb.users` file: **MAX_JOBS**, **JL/P**, **MAX_PEND_JOBS**

usergroup
Can be used with the following actions and keywords:

- *action* - addmember, rmmember, update, create, delete, set
- *value_pair* - the following keywords in the lsb.users file: **GROUP_ADMIN**, **GROUP_MEMBER**, **JL/P**, **MAX_JOBS**, **MAX_PEND_JOBS**, **PRIORITY**, **USER_SHARES**

host
Can be used with the following actions and keywords:

- *action* - update, add
- *value_pair* the following keywords in the lsb.hosts file: **MXJ**, **JL/U**, **EXIT_RATE**, **io**, **it**, **ls**, **mem**, **pg**, **r15s**, **r1m**, **r15m**, **swp**, **tmp**, **ut**
- *value_pair* - the following keywords in the lsf.cluster.*clustername* file: **model**, **type**, **resources**

hostgroup
Can be used with the following actions and keywords:

- *action* addmember, rmmember, update
- *value_pair* - the following keyword in the lsb.hosts file: **GROUP_MEMBER**

queue
Can be used with the following actions and keywords:

- *action* - addmember, rmmember, update
- *value_pair* - the following keywords in the lsb.queues file: **UJOB_LIMIT**, **PJOB_LIMIT**, **QJOB_LIMIT**, **HJOB_LIMIT**, **FAIRSHARE**

limit
Can be used with the following actions and keywords:

- *action* - addmember, rmmember, update, create, delete
- *value_pair* - the following keywords in the lsb.resources file: **QUEUES**, **PER_QUEUE**, **USERS**, **PER_USER**, **HOSTS**, **PER_HOST**, **PROJECTS**, **PER_PROJECT**, **SLOTS**, **SLOTS_PER_PROCESSOR**, **MEM**, **TMP**, **SWP**, **JOBS**, **RESOURCE**

gpool
Can be used with the following actions and keywords:

- *action* - addmember, rmmember, update
- *value_pair* - the following keyword in the lsb.resources file: **DISTRIBUTION**

serviceclass
Can be used with the following actions and keywords:

- *action* - create, delete, update, addmember, rmmember
- *value_pair* - the following keywords in the lsb.serviceclasses file: **ACCESS_CONTROL**, **DESCRIPTION**

object_name
The name of the existing object, or the object that is being created.

value_pair
The key (object attribute) and allowed values that are used in a **bconf** command request. The *value_pair* has the form *keyword=value*, and it uses the same keywords and syntax as in LSF configuration files. Not all LSF configuration keywords can be used with all actions.

Use a semicolon to separate multiple *value_pair* entries. Use a dash - or empty parentheses () to reset keywords to default values, depending on the keyword in the LSF configuration files.

For more information about allowed actions, objects, and keywords, use the help command **bconf -h** *action object*.

Examples

```
bconf -h addmember hostgroup
```

```
bconf addmember hostgroup=hgroupA "GROUP_MEMBER = host1"
```

```
bconf rmmember hostgroup=hgroupA "GROUP_MEMBER=host1 host2"
```

```
bconf update host=host1 "MXJ=10; JL/U=5"
```

```
bconf create usergroup=groupA "GROUP_MEMBER=(elaine tina toby); USER_SHARES=([elaine,10]
[default,5]); MAX_JOBS=500; MAX_PEND_JOBS=10000"
```

```
bconf rmmember queue=normal "FAIRSHARE=USER_SHARES[[joe, 10]]"
```

```
bconf addmember serviceclass=sla2 "ACCESS_CONTROL=QUEUES[normal]"
```

-c "*comment*"
Logs the text of *comment* as an administrator comment in the `liveconf.hist` file. The maximum length of the comment string is 512 characters. Embed the *comment* in double quotation marks, and do not include the new line character (\n).

-f
Disables interaction and forces **bconf delete** command requests to run without confirmation. Applies only to the `delete` action.

hist [-l|-w] [-o *object_type*] [-u *user_name*] [-T *time_period*] [-a *action*] [-f *config_file*] [*history_file*]
Queries the **bconf** command history file `liveconf.hist` located under the $LSB_SHAREDIR/*cluster_name*/`logdir` directory, or queries a specific history file (*history_file*). Output is filtered by the specified criteria. By default, only **bconf** command requests made by the current user are displayed.

-l
Long display format.

-w
Wide display format.

-o *object_type*
Displays entries that include the specified *object_type*. The following *object_type* values are supported: `user`, `usergroup`, `host`, `hostgroup`, `queue`, `limit`, `gpool`, `serviceclass`.

-u *user_name*
Displays entries for requests that are made by the specified *user*. To display **bconf** command requests from all users, specify the `-u all` option.

-T *time_period*
Displays entries within the specified time period. For syntax, see "Time Interval Format" in the **bhist** command reference.

-a *action*
Displays entries that include the specified *action*. The following *action* values are supported: `addmember`, `rmmember`, `update`, `create`, `add`, `delete`.

-f *config_file*
Displays entries that include the specified *config_file*. The following *config_file* values are supported: `lsb.resources`, `lsb.queues`, `lsb.users`, `lsb.hosts`, `lsf.cluster.`*clustername*, or `lsb.serviceclasses`.

history_file
Displays entries from the specified history file. By default, the history file is `liveconf.hist`.

disable
Blocks all **bconf** command requests until the next reconfiguration or restart of daemons with the **badmin reconfig**, **badmin mbdrestart**, or **lsadmin reconfig** commands (for manual changes to `lsf.cluster` file). Use the `disable` option before you change configuration manually files to make sure that you are editing files corresponding to the current configuration. Only the primary cluster administrator can disable live reconfiguration.

-h [*action* [*object_type*]]
Prints command usage to `stderr` and exits. Use for more information about allowed actions, objects, and the keywords that can be used with each object type.

bconf -h *action*
Lists allowed object types for the specified action.

bconf -h *action object_type*
Lists allowed value pairs for the specified *action* and *object_type*. If both *action* and *object_type* are specified, you can omit the `-h` option.

-pack
Reads multiple requests and sends them to `mbatchd` at the same time. **bconf** reads and parses the text file, with each line an individual **bconf** request. However, the requests are grouped together and sent to `mbatchd` at one time. If a line in the request fails, **bconf -pack** stops at that line.

-V
Prints LSF release version to `stderr` and exits.

bconf hist default output

The **bconf hist** command displays the **bconf** command events in shortened form, without comments or details of affected objects. Column content is truncated as required and marked with an asterisk (*).

TIME
Time of the **bconf** request.

OBJECT
The type of object specified.

NAME
The name of the object specified.

ACTION
Action that is performed on the object.

USER
User who made the **bconf** request.

IMPACTED_OBJ
All objects that are changed as a result of the **bconf** request.

```
bconf hist -u all
TIME                OBJECT    NAME   ACTION   USER   IMPACTED_OBJ
Nov 9 15:19:50 2010  limit     aaa    create   ellen  limit=aaa
Nov 9 15:19:46 2010  limit     aaa    update   leyang limit=aaa
Nov 9 15:19:37 2010  usergroup ug1    delete   ellen  queue=normal owners*
                                                      limit=bbb
                                                      usergroupr=ug1
Nov 9 15:19:28 2010  queue     normal update   leyang queue=normal
Nov 9 15:19:10 2010  host      host1  update   ellen  host=host1
```

bconf hist wide output (-w)

Wide output displays the same columns, but without truncating column contents.

```
bconf hist -w
TIME                OBJECT    NAME    ACTION   USER     IMPACTED_OBJ
Nov  9 15:19:50 2011 limit     aaa     create   ellen    limit=aaa
```

```
Nov  9 15:19:46 2011 limit     aaa     update   leyang   limit=aaa
Nov  9 15:19:37 2011 usergroup ug1     delete   ellen    queue=normal owners q1 q2 q3;
limit=bbb;
usergroup=ug1
```

bconf hist long output with the -l option

Long output displays all details of the requested **bconf** command events, including the new value of each affected object. Names of changed configuration files are included.

```
bconf hist -l
Mon Nov 18 15:19:45 2009: Limit <aaa> created by user <admin1> with requested values
<PER_HOST=all; RESOURCE=[A,5]; USERS=ug1 ug2 ug3> and comments <This is an example of a create
 action on a limit object named aaa.>
Changes made:
Limit <aaa> created in lsb.resources with <PER_HOST=all; RESOURCE=[A,5]; USERS=ug1 ug2 ug3>
------------------------------------------------------------
Mon Nov 18 15:19:45 2009: Usergroup <ug1> deleted by user <admin1> with comments <This is an
example of a delete action on a usergroup object named ug1.>
Changes made:
Usergroup <ug1> deleted in lsb.users
Limit <aaa> updated in lsb.resources with <USERS=ug2>
Queue <owners> updated in lsb.queues with <USERS=ug2 ug3>
------------------------------------------------------------
Mon Nov 18 15:19:45 2009: Queue <q1> updated by user <admin2> with requested values
<FAIRSHARE=USERSHARE[[ellen, 2]];QJOB_LIMIT=10> and comments <This is an example of an update
 action on a queue object named q1.>
Changes made:
Queue <q1> updated in lsb.queues with <QJOB_LIMIT=10>
------------------------------------------------------------
Mon Nov 18 15:19:45 2009: Limit <aaa> member added by user <admin2> with requested values
<USERS=julie> and comments <This is an example of an addmember action on a limit object named
aaa.>
Changes made:
Limit <aaa> updated in lsb.resources with <USERS=ellen user4 julie>
------------------------------------------------------------
Wed Jul 28 17:16:28 2010: Host <host78> added by user <usr9> with requested value <mem=500/100>
Changes made:
Host <host78> added in <lsf.cluster.x123> with <hostname=host78>
Host <host78> added in <lsb.hosts> with <HOST_NAME=host78; MXJ=!; mem=500/100>
------------------------------------------------------------
Wed Jul 28 17:17:08 2010: Host <host78> updated by user <usr9> with requested value
<mem=500/100>
Changes made:
Host <host78> updated in <lsb.hosts> with <mem=500/100>
```

Diagnostics

If the command ran correctly, the exit code is 0. A negative exit code the number of key-value pairs that contain errors.

See also

lsb.queues, lsb.hosts, lsb.resources, lsb.serviceclasses, lsb.users, lsf.cluster, lsf.conf

Chapter 8. bdata

Provides a set of subcommands to query and manage IBM Spectrum LSF Data Manager. If no subcommands are supplied, **bdata** displays the command usage.

Synopsis

LSF data management subcommands and option syntax synopsis

`bdata` *subcommand options*

`bdata` [`-h[elp]` | `-V`]

Subcommands

LSF data management subcommands and options

File-based cache query:

`bdata cache` [`-w` | `-l`] [`-u all` | `-u` *user_name*] [`-g all` | `-g` *user_group_name*] [`-dmd` *cluster_name*] [*host_name*`:`]*abs_file_path*

Job-based cache query:

`bdata cache` [`-dmd` *cluster_name*] [`-w` | `-l`] *job_ID*[`@`*cluster_name*]

Job-based cache query:

`bdata cache` [`-dmd` *cluster_name*] [`-w` | `-l`] *job_ID*[`@`*cluster_name*]

Change group after stage out:

`bdata chgrp` [`-dmd` *cluster_name*] `-g` *user_group_name* [*host_name*`:`]*abs_file_path*

Change group after stage in:

`bdata chgrp` [`-dmd` *cluster_name*] `-g` *user_group_name* `-tag` *tag_name*

Change the file permission mode of a file:

`bdata chmod` [`-dmd` *cluster_name*] `-mode` *octal_mode* `-mode` [*host_name*`:`]*abs_file_path*

Tag query:

`bdata tags list` [`-w`] [`-u all` | `-u` *user_name*] [`-dmd` *cluster_name*]

Tag cleanup:

`bdata tags clean` [`-u` *user_name*] [`-dmd` *cluster_name*] *tag_name*

Effective configuration query:

`bdata showconf`

Connections query:

`bdata connections` [`-w`]

Administration - reconfigure and shut down LSF data manager:

`bdata admin reconfig`

`bdata admin shutdown` [*host_name*]

Common options

-w
Wide format. Displays the information in a wide format. Use this option only with the **cache**, **connections**, and **tags list** subcommands.

-l
Long format. Displays additional information about LSF data management files. Use this option only with the **cache** subcommand.

-dmd *cluster_name*

Query the LSF data manager corresponding to the specified remote cluster. Use this option only with the **cache**, and **tags** subcommands.

-u all | -u *user_name*
Query files in the cache for the specified user. Use `-u all` or `-u` *user_name* with file-based **bdata cache** and **bdata tags list**. You can use only `-u` *user_name* with **bdata tags clean**.

cache

Queries the LSF data management cache.

Options

File-based query

`bdata cache` [`-w` | `-l`] [`-u all` | `-u` *user_name*] [`-g all` | `-g` *user_group_name*] [`-dmd` *cluster_name*] "[*host_name*`:`/]*abs_file_path*"

`bdata cache` [`-w` | `-l`] [`-u all` | `-u` *user_name*] [`-g all` | `-g` *user_group_name*] [`-dmd` *cluster_name*] "[*host_name*`:`/]*abs_folder_path*/[`*`]"

Job-based query

`bdata cache` [`-dmd` *cluster_name*] [`-w` | `-l`] *job_ID*[`@`*cluster_name*]

Description

File-based and folder-based query

Use the **bdata cache** *abs_file_path* or **bdata cache** "*abs_folder_path*/[`*`]" command to determine whether the files or folders that are required for your job are already staged in to the cache.

LSF data manager administrators can see information about staged-in files in the cache for all users (with the `-u all` option) or for a specific user (with the `-u` *user_name* option). The **CACHE_PERMISSIONS** parameter in the `lsf.datamanager` file determines which cache is accessible to non-administrator users.

CACHE_PERMISSIONS=user
Each user has a cache in the staging area. Ordinary users can request information only about their own cached files. **user** is the default.

CACHE_PERMISSIONS=all
The staging area is a single cache. All users can see all files in the cache.

CACHE_PERMISSIONS=group
Each UNIX group has cache in the staging area. By default, only users that belong to the same primary group can see the files for their group.

If **CACHE_PERMISSIONS=group** is specified, the `-g` option shows the cached files that belong to the specified user group.

If you specify a host name (*host_name*`:`*abs_file_path* or *host_name*`:`*abs_folder_path*/), the **bdata cache** command shows the files or folders that are staged in from the specified host. The path must match the **bjobs -data** command output exactly.

If a host name is not specified, the **bdata cache** command shows files that are staged in from the current local host.

Job-based query

Use the **bdata cache** *job_ID* command to show files that are referenced by the specified job ID. If a cluster name (with the @*cluster_name*option) is not specified with the job ID, the current cluster name is assumed.

Cache cleanup for input and output file records

You can use file-based query to see input file records until LSF data manager cleans up the job record and input files. After the job is finished and the grace period that is specified by the **CACHE_INPUT_GRACE_PERIOD** parameter in the `lsf.datamanager` file expires, LSF data manager cleans up the job record and input files cannot be queried.

You can use job-based query to see input file records only until those jobs finish (DONE or EXIT status).

You can query output file records until the following events occur:

- All of the output file records associated with the job have TRANSFERRED or ERROR status.
- And the grace period that is specified by the **CACHE_OUTPUT_GRACE_PERIOD** parameter expires for all files.

If both output and input job records exist, you can query the cache until all of these conditions are met.

Output: Default format

By default, the following information is shown for each file:

HASH
The hash key of the particular copy of the file.

STATUS
The status of the file.

NEW
LSF data manager received a requirement for the file, but a transfer job is not submitted for it yet.

STAGING
For input files, the file is requested but is not yet in the cache. For output files, the file is in the cache and is either waiting to be transferred out or is being transferred out.

TRANSFERRED
For input files, the file is in the cache. For output files, the transfer job for the file is complete.

ERROR
Output file transfer failed.

UNKNOWN
During recovery, it's possible that previously transferred files might show up as unknown for a short period while data manager recovers its state.

LINKED
If LSF data manager can directly access the required file in the cache, no transfer job is needed and the file is not copied into the cache. LSF data manager creates a symbolic link from the cache to the required file. The LINKED status shows that the file was symbolically linked.

REF_JOB
For file-based query only. List of job IDs of jobs that request the file. REF_JOB is not displayed for job-based query.

XFER_JOB
The job ID of the transfer job. If LSF data manager can directly access the required file in the cache, no transfer job is needed and the file is not copied into the cache. A dash (-) indicates that no transfer job is associated with the file.

GRACE
After files are no longer needed by any job, unused input and output files in the data manager cache are cleaned up after a configurable grace period (**CACHE_INPUT_GRACE_PERIOD** and **CACHE_OUTPUT_GRACE_PERIOD** parameters in `lsf.datamanager`). GRACE shows the remaining hours and minutes of the grace period.

- Input file records enter grace period after file transfer is complete (STATUS is TRANSFERRED), and the list of jobs for REF_JOB becomes empty. After the grace period expires, the files are cleaned up and can no longer be queried by file name. The default input grace period is 1440 minutes (one day).
- Output file records enter grace period immediately after their status becomes TRANSFERRED. However, the files and job records are not cleaned up until the grace periods expire for all stage-out requirements that are associated with the same job. Output files can be queried by file name until the grace period expires for all output file records associated with the job. The default output grace period is 180 minutes (3 hours). Files that are uploaded to the cache with the **bstage out -tag** command must be cleaned manually with the **bdata tags clean** command.

Output: long format

In a long format display, the following additional information is displayed:

PERMISSION
Access permissions for the file, which is defined by the **CACHE_PERMISSIONS** parameter in `lsf.datamanager`.

When **CACHE_PERMISSIONS=all**, the PERMISSION field shows `all`.

When the **CACHE_ACCESS_CONTROL=Y** parameter is configured in `lsf.datamanager`, the PERMISSION field shows the user group and the file permissions.

SIZE

Units for file size.

- *nnn* B if file size is less than 1 KB
- *nnn*[.*n*] KB if file size is less than 1 MB
- *nnn*[.*n*] MB if file size is less than 1 GB
- *nnn*[.*n*] GB if file size is 1 GB or larger
- *nnn*[.*n*] EB is displayed if file size is 1 EB or larger

MODIFIED
The last modified time of the file, as it was at job submission time or at the time of the stage out request.

CACHE_LOCATION
The full location of the file in the cache, as mounted on the data manager hosts.

```
bdata cache -l hostA:/home/user1/job.sh
------------------------------------------------------------------------------
INPUT:
hostA:/home/user1/job.sh

PERMISSION          user:user1
HASH                7fb71a04569b51c851122553e2c728c5
SIZE                5 MB
STATUS              TRANSFERRED
REF_JOB             1435@cluster1
XFER_JOB            1906@cluster2 FINISHED Mon Aug 18 09:05:25 2014
GRACE               -
MODIFIED            Thu Aug 14 17:01:57 2014

CACHE_LOCATION:
/scratch/user1/staging/stgin/user/user1/hostA/home/user1/job.sh/e2cc059b47c094544791664a51489c8c
```

Examples: query by file or folder

The file is in the cache with one shared copy that is cached for different jobs:

```
bdata cache hostA:/home/user1/transfer_tool.sh
------------------------------------------------------------------------------
INPUT:
hostA:/home/user1/transfer_tool.sh
HASH     STATUS   REF_JOB        XFER_JOB       GRACE
ab7dc9*  STAGING  2947@cluster1  2949@cluster1  -
                  2952@cluster1
                  2954@cluster1
```

The following job requests a file that is owned by the user group `design1`:

```
bsub -data /home/user1/data/file1.txt -datagrp design1 sleep 9999
Job <11297> is submitted to default queue <normal>.
```

Use the `-g` option to query files that belong to the specified group:

```
bdata cache -g designl /home/user1/data/file1.txt
------------------------------------------------------------------------------
INPUT:
hosta:/home/user1/data/file1.txt

HASH    STATUS       REF_JOB               XFER_JOB              GRACE
fbea85* LINKED       11297@cluster1        -                     -
```

Note: The status of the file `/home/user1/data/file1.txt` is `LINKED`, and `XFER_JOB` is shown as a dash (`-`). LSF data manager can directly access the required file in the cache, so no transfer job is needed and the file is not copied into the cache. LSF data manager created a symbolic link from the cache to the required file. The `LINKED` status shows that the file was symbolically linked. A dash (`-`) indicates that no transfer job is associated with the file.

When the input path ends in a slash (/), information about folders that were recursively copied into the cache are displayed.

```
bdata cache "/home/user1/folder1/" -l
------------------------------------------------------------------------------
INPUT:
hb05b10:/home/user1/folder1/

PERMISSION          group:lsf    rwx------    [manual]
HASH                eb72d80f6deeeaf51e7f2913451bb9da
SIZE                4 KB
STATUS              TRANSFERRED
REF_JOB             44843@lsf913
XFER_JOB            44844@lsf913 FINISHED Wed May 10 14:51:47 2017
GRACE               -
MODIFIED            Tue May  9 09:53:32 2017

CACHE_LOCATION:
/home/user1/scratch/staging/stgin/all/hosta/home/user1/folder1//eb72d80f6deeeaf51e7f2913451bb9da
```

When the input path ends in a slash and an asterisk (/*) only the top-level folder was requested.

```
bdata cache "/home/user1/folder1/*" -l
------------------------------------------------------------------------------
INPUT:
hb05b10:/home/user1/folder1/*

PERMISSION          group:lsf    rwx------    [manual]
HASH                f63ec6dc849a7390fc622620d88129f3
SIZE                4 KB
STATUS              TRANSFERRED
REF_JOB             44845@lsf913
XFER_JOB            44846@lsf913 FINISHED Wed May 10 14:51:48 2017
GRACE               -
MODIFIED            Tue May  9 09:53:32 2017

CACHE_LOCATION:
/home/user1/scratch/staging/stgin/all/hb05b10/home/user1/folder1/*/
f63ec6dc849a7390fc622620d88129f3
```

When you use the asterisk character (*) at the end of the path, the data requirements string must be in quotation marks.

Examples: query by job

The job requests two input files. During job execution, file data2 is copied to two other locations. Files being staged out are listed as OUTPUT and show their destinations:

```
bdata cache 84044
Job <84044@cluster1> has the following file records in LSF data manager:
------------------------------------------------------------------------------
INPUT:
hostA:/home/user1/data2

HASH    STATUS       XFER_JOB               GRACE
68990b* TRANSFERRED  84045@cluster1         -

------------------------------------------------------------------------------
INPUT:
hostA:/home/user1/data3

HASH    STATUS       XFER_JOB               GRACE
e2fff4* TRANSFERRED  84056@cluster1         -

------------------------------------------------------------------------------
OUTPUT:
hostB:/home/user1/data2
TO:
hostA:/scratch/user1/workspace

HASH    STATUS       XFER_JOB               GRACE
68990b* TRANSFERRED  84091@cluster1         -
```

Examples: query a single file, long output

```
bdata cache -l hostA:/home/user1/testDATA/rt/rt1/ada2 -u user1
------------------------------------------------------------------------------
INPUT:
hostA:/home/user1/testDATA/rt/rt1/ada2

PERMISSION          user:user1
HASH                7fb71a05130b51c673953948e2c397c5
SIZE                50 MB
STATUS              TRANSFERRED
REF_JOB             1435@cluster1
XFER_JOB            1906@cluster1 FINISHED Mon Aug 18 09:05:25 2014
GRACE               -
MODIFIED            Wed Apr 30 14:41:22 2014

CACHE_LOCATION:
/data/cache/stgin/user/user1/hostA/home/user1/testDATA/rt/rt1/ada2/7fb71a05130b51c673953948e2c397c5
```

The following example uses the -g option to query a file that belongs to the user group design1:

```
bdata cache -l -g design1 /home/user1/data/file1.txt
------------------------------------------------------------------------------
INPUT:
hosta:/home/user1/data/file1.txt

PERMISSION          group:design1
HASH                fbea858bdf6ddefc6c7f44dc6a08f1a6
SIZE                4 B
STATUS              LINKED
REF_JOB             11297@cluster1
XFER_JOB            -
GRACE               -
MODIFIED            Thu Oct  9 07:54:19 2014

CACHE_LOCATION:
/scratch/data/user1/staging1/stgin/group/design1/hosta/home/user1/data/file1.txt/
fbea858bdf6ddefc6c7f44dc6a08f1a6
```

Examples: query multiple files for the same job, long output

```
bdata cache -l 1909
Job <1909@cluster1> has the following file records in LSF data manager:
------------------------------------------------------------------------------
INPUT:
hostA:/home/user1/testDATA/status.1

PERMISSION          user:user1
HASH                0f9267a79de4bb2f9143b61ab741afda
SIZE                290 B
STATUS              TRANSFERRED
XFER_JOB            1908@cluster1 FINISHED Mon Aug 18 10:01:51 2014
GRACE               -
MODIFIED            Thu Jul  3 10:50:53 2014

CACHE_LOCATION:
/data/cache/stgin/user/user1/hostB/home/user1/testDATA/status.1/0f9267a79de4bb2f9143b61ab741afda

------------------------------------------------------------------------------
INPUT:
hostA:/home/user1/testDATA/status.2

PERMISSION          user:user1
HASH                2b992669d4ce96902cd639dda190a586
SIZE                0 B
STATUS              TRANSFERRED
XFER_JOB            1910@cluster1 FINISHED Mon Aug 18 10:02:27 2014
GRACE               -
MODIFIED            Thu Jul  3 10:49:36 2014

CACHE_LOCATION:
/data/cache/stgin/user/user1/hostB/home/user1/testDATA/status.2/2b992669d4ce96902cd639dda190a586

------------------------------------------------------------------------------
OUTPUT:
hostA:/home/user1/testDATA/status.1
TO:
hostA:/scratch/user1/data/out
HASH                0f9267a79de4bb2f9143b61ab741afda
SIZE                290 B
STATUS              STAGING
XFER_JOB            1911@cluster1
GRACE               -
MODIFIED            Thu Jul  3 10:50:53 2014

CACHE_LOCATION:
/data/cluster1cache/stgout/cluster1/hostA/1909/home/user1/testDATA/
status.1/0f9267a79de4bb2f9143b61ab741afda

------------------------------------------------------------------------------
OUTPUT:
hostA:/home/user1/testDATA/status.2
TO:
hostA:/scratch/user1/data/out

HASH                2b992669d4ce96902cd639dda190a586
SIZE                0 B
STATUS              STAGING
XFER_JOB            1912@cluster1
GRACE               -
MODIFIED            Thu Jul  3 10:49:36 2014

CACHE_LOCATION:
/data/cache/stgin/user/user1/hostB/home/user1/testDATA/status.2/2b992669d4ce96902cd639dda190a586
```

Examples: query the same file for multiple jobs, long output

```
bdata cache -l /home/user1/testDATA/status.1
------------------------------------------------------------------------------
INPUT:
hostA:/home/user1/testDATA/status.1

PERMISSION          user:user1
HASH                0f9267a79de4bb2f9143b61ab741afda
SIZE                290 B
STATUS              TRANSFERRED
REF_JOB             1909@cluster1
                    1913@cluster1
```

```
XFER_JOB            1908@cluster1 FINISHED Mon Aug 18 10:01:51 2014
GRACE               -
MODIFIED            Thu Jul  3 10:50:53 2014

CACHE_LOCATION:
/data/cluster1cache/stgin/user/user1/hostA/home/user1/testDATA/status.1/0f9267a79de4bb2f9143b61ab741afda
```

Examples: query by cluster with the -dmd option

The following example queries all instances of the file /newshare/scal/user1/data_files/seqdata.0 on host hostA for cluster cluster1.

```
bdata cache -dmd cluster1 hostA:/newshare/scal/user1/data_files/seqdata.0
-------------------------------------------------------------------------------
INPUT:
hostA:/newshare/scal/user1/data_files/seqdata.0

HASH    STATUS       REF_JOB               XFER_JOB              GRACE
6e91e3* TRANSFERRED  15@cluster1           5@cluster1            -
                     16@cluster1
```

The following example queries data requirements for job 15 on cluster cluster1.

```
bdata cache -dmd cluster1 15
Job <15@cluster1> has the following file records in LSF data manager:
-------------------------------------------------------------------------------
OUTPUT:
hostA:/newshare/scal/user1/data_files/15/seqdata.1
TO:
hostB:/newshare/scal/user1/data_files/seqdata.15

HASH    STATUS       XFER_JOB              GRACE
e21557* STAGING      5@cluster1           -
```

Examples: file query when user group cache access is enabled

When the **CACHE_ACCESS_CONTROL=Y** parameter is configured in lsf.datamanager, the bdata cache -l command shows the user group and the file permissions.

```
bdata cache -l 1152
Job <1152@dm1> has the following file records in LSF data manager:
-------------------------------------------------------------------------------
INPUT:
hostA:/newshare/scal/user1/data_files/15/seqdata.1
PERMISSION          group:pcl   rwxr-x---   [manual]
HASH                b7202f200c0240a66493f81f0e2e8875
SIZE                1 KB
STATUS              TRANSFERRED
XFER_JOB            1153@dm1 FINISHED Tue Nov 17 10:42:25 2015
GRACE               -
MODIFIED            Tue Apr 19 15:59:28 2015

CACHE_LOCATION:
/data/cluster1cache/stgin/user/user1/hostA/home/user1/testDATA/seqdata.1/b7202f200c0240a66493f81f0e2e8875
```

chgrp

Changes the user group of a file in the LSF data management cache.

Options

Change user group after stage out.

bdata chgrp [-dmd *cluster_name*] -g *user_group_name* "[*host_name*:]*abs_file_path*"

bdata chgrp [-dmd *cluster_name*] -g *user_group_name* "[*host_name*:]*abs_folder_path*/[*]"

Change user group after stage in.

bdata chgrp [-dmd *cluster_name*] -g *user_group_name* -tag *tag_name*

Description

Note: The **CACHE_ACCESS_CONTROL = Y** parameter must be configured in the `lsf.datamanager` file to use the **bdata chgrp** command. Only the owner of the file, folder, or data tag can change the group access with the **bdata chgrp** command.

Change group after stage out

Use the **bdata chgrp -g** *group_name* command and specify the name of a data file or folder that is already staged in to the cache. The **chgrp** command changes the user group of a transferred data file and the directory it is in. The file must be in the transferred state or idle state for the command to succeed.

If you specify a folder, either recursive (`abs_folder_path/`) or non-recursive (`abs_folder_path/*`), the folder must be in the transferred state or idle state. The user group of both the folder and its contents are changed. The top-level folder permissions are set to `0770` (`rwxrwx---`). The permissions of other directories under the top-level directory are set to `0750` (`rwxr-x---`).

When you use the asterisk character (`*`) at the end of the path, the data requirements string must be in quotation marks.

If you specify an optional host name (*host_name*`:`*abs_file_path*), the **bdata chgrp** command changes the user group of the files or folders that are staged in from the specified host.

Change group after stage in

Use the **bdata chgrp -g** *group_name* command and specify a tag name to change the group of all the files in the cache that is associated with the tag to the group name specified by *group_name*. The group of the tag directory and its contents is changed. The command sets the permissions of the top-level tag directory to `0770` (`rwxrwx---`). The permission of other directories in the top-level tag directory is set to `0750` (`rwxr-x---`).

The command is equivalent to the **chmod ug+r, o-rwx** command, and sets the owner and group read permissions of all files (not directories).

chmod

Changes the file permission mode of a file that is staged in to the LSF data management cache.

Options

`bdata chmod` [`-dmd` *cluster_name*] `-mode` *octal_mode* `-mode` [*host_name*`:`]*abs_file_path*

Description

Note: The **CACHE_ACCESS_CONTROL = Y** parameter must be configured in the `lsf.datamanager` file to use the **bdata chmod** command. Only the owner of the file or data tag can change the group access with the **bdata chmod** command.

Use the **bdata chmod** command to change the permission mode of files that are required for your job that are already staged in to the cache.

The file must be in the transferred or idle state for the **chmod** command to succeed.

The **chmod** command interacts with the **PERMISSION_MASK** parameter in the `lsf.datamanager` file. The permission mask controls which permissions on a file can be changed by the **bdata chmod** command.

PERMISSION_MASK=750
File owners can modify user `rwx` bits, group `rw` bits, but not global bits.

PERMISSION_MASK=777
File owners can modify all the bits.

PERMISSION_MASK=000
FIle owners cannot modify any permissions.

The permissions on the directory that the file is in are changed to be consistent with the file permission, but the owner always has full access at the directory level. If the permissions on the file allow the group or others permission to read, write, or execute, the same permissions are enabled on the directory. For example, if **PERMISSION_MASK=777** and the file mode is 570 (`r-xrwx---`), the permissions on the directory are changed to 770 (`rwxrwxr---`).

The value of the *octal-mode* argument is three octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. The first digit selects permissions for the user who owns the file: read (4), write (2), and execute (1). The second digit selects the permissions for other users in the group that is associated with the file. The third digit selects the permissions for other users not in the group that is associated with the file.

tags

LSF data management tag query and cleanup options

Options

Tag query:

`bdata tags list` [`-w`] [`-u all` | `-u` *user_name*] [`-dmd` *cluster_name*]

Tag cleanup:

`bdata tags clean` [`-u` *user_name*] [`-dmd` *cluster_name*] *tag_name*

Description

The **`list`** command displays the names of the known tags.

When the **CACHE_ACCESS_CONTROL=Y** parameter is configured in `lsf.datamanager`, the **`list`** command displays the user group that the file belongs to. All users (not just the administsrator) can specify the –u option to see tags that are owned by other users.

```
bdata tags list
NAME      USER    GROUP     LAST_UPLOAD          LAST_DOWNLOAD
code      user1   pcl       06/12/2015 15:50:10  -
tag1      user1   <error>   03/12/2015 13:51:59  -
tag2      user2   test      03/12/2015 13:52:06  -
tag3      user2   test      03/12/2015 13:52:09  -
```

If a **`bdata chgrp`** command did not successfully change the user group of the file, the GROUP column shows `<error>`.

A dash (`-`) in the GROUP column of a tag query indicates that **CACHE_ACCESS_CONTROL=Y** is not configured.

The **`clean`** command deletes the tag and all its containing files. Data manager administrators can clean any tags. Ordinary users can clean only their own tags. Each LSF data manager manages its own cache area. You can query and manage tags through `-dmd` *cluster_name* option, which redirects the request to the **dmd** that serves the specified cluster.

Examples

Query known local tags:

```
bdata tags list -u all
NAME       USER   LAST_UPLOAD           LAST_DOWNLOAD
tag_name1  user1  06/27/2015 16:37:52   06/28/2015 16:37:52
tag_name2  user2  06/23/2015 16:37:52   06/28/2015 16:37:52
tag_name3  user3  06/25/2015 16:37:52   06/30/2015 16:37:52
...
```

A dash (-) in the LAST_UPLOAD column of a tag query indicates tags that were created, but not finished their first upload:

```
bdata tags list -u all
NAME       USER   LAST_UPLOAD       LAST_DOWNLOAD
tag_name1  user1  -                 -
tag_name2  user2  -                 -
tag_name3  user3  -                 -
...
```

In this state, the scheduler does not schedule the job to use the tags until the first upload to the staging area is finished and the LAST_UPLOAD time is set. If the upload fails, the tag stays this way. Use **bdata tags clean** to clean up the tags for failed uploads.

showconf

LSF data management effective configuration query option

`bdata showconf`

List the effective values of LSF data manager configuration parameters from `lsf.datamanager` and `lsf.conf`.

Example

Query the current LSF data manager configuration:

```
 bdata showconf
LSF data management configuration at Wed Aug 20 16:03:40 2014
        ADMINS = lsfadmin admin1
        CACHE_INPUT_GRACE_PERIOD = 1440 (minutes)
        CACHE_OUTPUT_GRACE_PERIOD = 100 (minutes)
        CACHE_PERMISSIONS = group
        FILE_TRANSFER_CMD = /usr/bin/scp
        LSB_TIME_DMD = 0
        LSF_DATA_HOSTS = hostA hostB
        LSF_DATA_PORT = 39486
        LSF_LOGDIR = /opt/lsf/LSF913/cluster1/log
        LSF_LOG_MASK = LOG_WARNING
        CACHE_ACCESS_CONTROL = Y
        QUERY_NTHREADS = 4
        STAGING_AREA = nfs1:/mnt/zfsvol/hostC/user1/LSF913/cluster1
```

connections

Query LSF data management connections. Lists currently connected **mbatchd** daemons, with master LSF data manager host names, their status, and the outgoing and incoming connections for remote LSF data managers.

`bdata connections [-w]`

Output

The **bdata connections** output has the following sections.

The LOCAL section displays two tables:

- The first table displays the configuration of the **LSF_DATA_HOSTS** and **LSF_DATA_PORT** parameters for the local LSF data manager. The currently active LSF data manager master host name is marked with an asterisk ([*]).
- The second table displays the connection status of registered **mbatchd** daemons.

  CLUSTER
  Displays the cluster name.

  MASTER
  Displays the name of the LSF cluster master host that is connected to this data manager.

STATUS
Displays the connection status for the cluster (ok or disc).

The OUTGOING REMOTE section displays the connection status of remote LSF data managers for stage out jobs that are configured in the RemoteDataManagers section of the lsf.datamanager file:

CLUSTER
Corresponds to the **CLUSTER** column in the lsf.datamanager file, and displays the name of the remote LSF cluster.

LSF_DATA_HOSTS
Displays the data hosts configured in the RemoteDataManagers section of the lsf.datamanager file. The currently active LSF data manager master host name is marked with an asterisk ([*]).

LSF_DATA_PORT
Displays the LSF data manager connection port that is specified by the **LSF_DATA_PORT** parameter in the lsf.conf file.

STATUS
Displays the connection status (ok or disc).

The INCOMING REMOTE section displays the status of remote LSF data managers that connect to the local LSF data manager to query for file availability.

CLUSTER
Corresponds to the **CLUSTER** column in the lsf.datamanager file, and displays the name of the remote LSF cluster.

DMD_MASTER
Displays the connected LSF data manager master host name for the cluster.

LSF_DATA_PORT
Displays the LSF data manager connection port that is specified by the **LSF_DATA_PORT** parameter in the lsf.conf file.

STATUS
Displays the connection status (ok or disc).

If the RemoteDataManagers section is configured in the lsf.datamanager file in the local cluster, the OUTGOING REMOTE section is displayed. If the RemoteDataManagers section is configured in the lsf.datamanager file in the remote cluster, the local cluster displays the INCOMING REMOTE section. Only the cluster that is sending jobs needs to configure the RemoteDataManagers section.

Example

Query LSF data manager connections:

```
bdata connections
LOCAL: (LSF_DATA_PORT=2960)
LSF_DATA_HOSTS
[*]hostB

CLUSTER             MASTER              STATUS
cluster1            hostB               ok

OUTGOING REMOTE:
CLUSTER             LSF_DATA_HOSTS      LSF_DATA_PORT  STATUS
hamburg             [*]dmd1.hamburg     4104           ok
toulouse            dmd2.toulouse       12058          disc

INCOMING REMOTE:
CLUSTER             DMD_MASTER          LSF_DATA_PORT  STATUS
waterloo            dmd3.waterloo       33994          ok
venice              dmd5.venice         13514          ok
```

admin

Reconfigure and shut down the LSF data manager daemon (**dmd**).

Options

Reconfigure

`bdata admin reconfig`

Shut down

`bdata admin shutdown` [*host_name*]

Description

Only the LSF data manager administrator or root can run these commands.

Use the **reconfig** subcommand after you change configuration in the `lsf.datamanager` file. The configuration files are checked before **dmd** is reconfigured. If the configuration is not correct, reconfiguration does not start.

The **shutdown** subcommand kills the `lsf.datamanager` data manager daemon processes on the specified host. The request is rejected if the LIM on the node is running.

To shut down the `lsf.datamanager` data manager, you must first shut down LIM with the **lsadmin limshutdown** command.

If a host name is not specified, `lsf.datamanager` data manager attempts to shut down all candidate `lsf.datamanager` data manager hosts.

The output of the **bdata admin** command is similar in format to the **lsadmin limshutdown** commands and the **lsadmin reconfig** and **badmin hshutdown** and **badmin reconfig** commands.

Examples

Reconfiguration

```
bdata admin reconfig
Checking configuration files ...
No errors found.

LSF data manager daemon (dmd) on <hostA> is reconfiguring ... Initiated.
```

Shutdown

```
bdata admin shutdown
LSF data manager daemon (dmd) on <hostA> is shutting down... Done.
LSF data manager daemon (dmd) on <hostB> is shutting down... Host unreachable.
LSF data manager daemon (dmd) on <hostc> is shutting down... Cannot shut down LSF data
manager daemon (dmd) when the local lim is running.
```

Help and version options

IBM Spectrum LSF Data Manager help and version display options

`bdata` [`-h[elp]` | `-V`]

-h[help]
Displays the command usage of the **bdata** command to `stderr` and exits.

-V
Prints the IBM Spectrum LSF Data Manager release version to `stderr` and exits.

See also

bhist, **bjobs**, **bmod**, **bstage**, **bsub**, `lsf.conf`, `lsf.datamanager`

Chapter 9. bentags

Queries or removes information about the energy policy tag from the **mbatchd** daemon, which is saved in the energy-aware scheduling database. Used with energy policy, or energy aware scheduling feature.

Synopsis

`bentags` [`-o` *cpu_frequency* | `-o` *time* | `-o` *energy*] [`-u` *user* | `-u all`] [*energy_tag*...]

`bentags -r` [`-u` *user* | `-u all`] [*energy_tag*...]

`bentags` [`-h` | `-V`]

Description

The energy policy tag is the unique identifier of a job's energy data, used to identify the predicted energy usage, predicted run time of the job, and the job performance degradation. The energy policy tag name is specified by using `esub.eas`. When you submit a job, make sure that the energy policy tag name is unique for each of your jobs.

The **bentags** command displays all the energy tag names that are generated by the user. When you remove an energy tag, specify the tag name or user name. If you specify the user name, all tags that belong to that user are removed. Only an LSF administrator and root can remove another user's energy policy tags.

Options

-h
Provides extended help information.

-V
Displays the name of the command, release number, service level, service level date, and lowest level of the operating system to run this release.

-o *cpu_frequency* | *time* | *energy*
Sort the energy tag's data by *cpu_frequency* / *time* / *energy*, in descending order. Default is to sort by CPU frequency. *cpu_frequency* matches to CPU_FREQ, *time* matches to EST_RUNTIME, and *energy* matches to EST_USAGE.

-r
Remove energy policy tag. This option must be followed by the –u option or an energy tag name.

-u *user_name*... | -u all
Displays only the energy policy tags that are submitted by the specified user. The keyword `all` specifies all users. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

energy_tag
Display the data of a specific tag name.

Output: Energy tag data

Energy Tag Name
The energy tag name.

Job ID
The identifier of the job that generated the data.

User
The user who generated the data.

Default Frequency
The CPU frequency that is used when the energy tag is generated.

Node's Energy Use
The direct current (DC) energy consumption per node of the job at the default frequency.

Runtime
The execution time of the job at the default frequency.

CPU_FREQ (GHz)
The CPU frequency.

EST_USAGE (kWh)
The job's estimated DC energy consumption per host, at this frequency.

ENER_VAR (%)
The percentage of variation in DC energy that is consumed at this frequency. A negative value indicates the percentage of energy savings and a positive value indicates percentage of additional energy consumed.

EST_RUNTIME (Seconds)
The estimated run time of the job at this frequency.

RUNTIME_VAR (%)
The percentage of performance variation at this frequency. A negative value indicates the performance improvement and a positive value indicates the performance degradation.

POWER (W)
The power, measured in watts (W).

Example: Show output for tag long_running_job

Use **bentags long_running_job** to view data for the energy tag *long_running_job*:

```
bentags long_running_job
                Energy Tag Name: long_running_job
                       Job ID: 526
                         User: UserA
            Default Frequency: 2.00 GHz
            Node's Energy Use: 0.007707 kWh
                              Runtime: 147 Seconds
CPU_FREQ(GHz)   EST_USAGE(kWh)   ENER_VAR(%)  EST_RUNTIME(Sec)  RUNTIME_VAR(%)   POWER(W)
2.30              0.007582        -1.62            136              -7.48          200.70
2.20              0.007859         1.93            138              -6.12          205.03
2.10              0.007418        -3.75            140              -3.06          190.74
2.00              0.007707         0.00            147               0.00          188.74
1.90              0.007308        -5.18            144              -2.04          182.70
1.80              0.007391        -4.09            148               0.68          179.78
1.70              0.007310        -5.15            148               0.68          177.81
1.60              0.007303        -5.24            149               1.36          176.45
1.50              0.007243        -6.01            150               2.04          173.83
```

Example: Remove UserA.long_running_job energy policy tag from database

Use **bentags -r UserA.long_running_job** to remove UserA's *long_running_job* energy policy tag from the database.

```
bentags -r UserA.long_running_job
The energy policy tag <UserA.long_running_job> is removed.
```

Example: Remove all energy policy tags for UserA

Use **bentags -r -u UserA** to remove all of UserA's energy policy tags from the database.

```
bentags -r  -u UserA
The energy policy tag <UserA.long_running_job> is removed.
The energy policy tag <UserA.short_running_job> is removed.
The energy policy tag <UserA.medium_running_job> is removed.
```

See also

bjobs -l

Chapter 10. bgadd

Creates job groups

Synopsis

bgadd [-L *limit*] [-sla *service_class_name*] *job_group_name*

bgadd [-h | -V]

Description

Creates a job group with the job group name specified by *job_group_name*.

You must provide full group path name for the new job group. The last component of the path is the name of the new group to be created.

You do not need to create the parent job group before you create a sub-group under it. If no groups in the job group hierarchy exist, all groups are created with the specified hierarchy.

Options

-L *limit*

Specifies the maximum number of concurrent jobs allowed to run under the job group (including child groups). The -L option limits the number of started jobs (jobs in RUN, SSUSP, or USUSP state) under the job group. Specify a positive number between 0 and 2147483647. If the specified limit is zero (0), no jobs under the job group can run.

You cannot specify a limit for the root job group. The root job group has no job limit. Job groups added with no limits specified inherit any limits of existing parent job groups. The -L option only limits the lowest level job group created.

If a parallel job requests 2 CPUs (**bsub -n 2**), the job group limit is per job, not per slots used by the job.

By default, a job group has no job limit. Limits persist across **mbatchd** restart or reconfiguration.

-sla *service_class_name*

The name of a service class defined in lsb.serviceclasses, or the name of the SLA defined in the **ENABLE_DEFAULT_EGO_SLA** parameter in the lsb.params file. The job group is attached to the specified SLA.

job_group_name

Full path of the job group name. Job group names can be up to 512 characters long.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

Examples

- Create a job group named risk_group under the root group /:

```
bgadd /risk_group
```

- Create a job group named portfolio1 under job group /risk_group:

```
bgadd /risk_group/portfolio1
```

See also

bgdel, **bjgroup**

Chapter 11. bgdel

Deletes job groups

Synopsis

`bgdel [-u` *user_name* `| -u all]` *job_group_name* `| 0`

`bgdel -c` *job_group_name*

`bgdel [-h | -V]`

Description

Deletes a job group with the job group name specified by *job_group_name* and all its subgroups.

You must provide full group path name for the job group to be deleted. Deletion only takes effect after all jobs belonging to the group are cleaned out of **mbatchd** memory after the clean period.

Users can delete only their own job groups. LSF administrators can delete any job groups.

Job groups can be created explicitly or implicitly:

- A job group is created explicitly with the **bgadd** command.
- A job group is created implicitly by the the **bsub -g** or **bmod -g** command when the specified group does not exist. Job groups are also created implicitly when a default job group is configured (the **DEFAULT_JOBGROUP** in the `lsb.params` file, or the **LSB_DEFAULT_JOBGROUP** environment variable).

Options

0

Delete empty job groups. These groups can be explicit or implicit.

-u ***user_name***

Delete empty job groups owned by the specified user. Only administrators can use this option. These groups can be explicit or implicit. If you specify a job group name, the `-u` option is ignored.

-u all

Delete empty job groups and their sub groups for all users. Only administrators can use this option. These groups can be explicit or implicit. If you specify a job group name, the `-u` option is ignored.

-c ***job_group_name***

Delete all empty groups below the requested *job_group_name* including the *job_group_name* itself. These groups can be explicit or implicit.

job_group_name

Full path of the job group name.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Example

The following example deletes the job group /risk_group and all its subgroups:

```
bgdel /risk_group
Job group /risk_group is deleted.
```

See also

bgadd, **bjgroup**

Chapter 12. bgmod

Modifies job groups

Synopsis

bgmod [-L *limit* | -Ln] [-u *job_group_owner*] *job_group_name*

bgmod [-h | -V]

Description

Modifies the job group with the job group name that is specified by *job_group_name*.

Only root, LSF administrators, the job group creator, or the creator of the parent job groups can use the **bgmod** command to modify a job group limit.

You must provide full group path name for the modified job group. The last component of the path is the name of the job group to be modified.

Options

-L *limit*

Changes the limit of *job_group_name* to the specified *limit* value. If the job group has parent job groups, the new limit cannot exceed the limits of any higher-level job groups. Similarly, if the job group has child job groups, the new value must be greater than any limits on the lower-level job groups.

limit specifies the maximum number of concurrent jobs that are allowed to run under the job group (including child groups) -L limits the number of started jobs (RUN, SSUSP, USUSP) under the job group. Specify a positive number 0 - 2147483647. If the specified limit is zero, no jobs under the job group can run.

You cannot specify a limit for the root job group. The root job group has no job limit. The -L option limits only the lowest level job group specified.

If a parallel job requests 2 CPUs (**bsub -n 2**), the job group limit is per job, not per slots used by the job.

-Ln

Removes the existing job limit for the job group. If the job group has parent job groups, the job modified group automatically inherits any limits from its direct parent job group.

-u *job_group_owner*

Changes the job group owner to the specified *job_group_owner* user name.

To specify a Windows user account, include the domain name in uppercase letters and use either a single backslash (*DOMAIN_NAME\user_name*) from a Windows command prompt or a double backslash (*DOMAIN_NAME\\user_name*) from a UNIX command line.

Use of this option is restricted to the following users:

- root
- Primary LSF administrator
- Parent job group owner
- Current job group owner

job_group_name

Full path of the job group name.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

The following command modifies only the limit of group `/canada/projects/test1`. It does not modify limits of `/canada` or `/canada/projects`.

```
bgmod -L 6 /canada/projects/test1
```

To modify limits of `/canada` or `/canada/projects`, you must specify the exact group name:

```
bgmod -L 6 /canada
```

Or

```
bgmod -L 6 /canada/projects
```

See also

bgadd, **bgdel**, **bjgroup**

Chapter 13. bgpinfo

Displays information about global fairshare.

Synopsis

`bgpinfo` *subcommand*

`bgpinfo` [`-h` | `-V`]

Description

The **bgpinfo** command provides a set of subcommands to get information about global fairshare.

This command can also display remote fairshare load. The remote fairshare load impacts your dynamic priority for job scheduling.

Information about each subcommand is available through the help command.

Subcommand synopsis

`conf` [`-l`] [*global_policy_name*...]

`fsload` [`-l` [-c *cluster_name*]] [*global_policy_name*...]

`status` [`-l` [*global_policy_name* ...]]

`-h`

`-V`

Options

subcommand

Runs the specified subcommand. See Usage section.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Usage

conf [-l] [global_policy_name ...]

Shows summary of global fairshare policy configurations.

POLICY_NAME

The name of the global policy.

TYPE

Type of global policy.

If *global_policy_name* is specified, shows only the information for the specified global policies.

-l

Long format. Shows the following additional information:

SYNC_INTERVAL

Fairshare load synchronization interval.

SYNC_MODE

Synchronization mode (all or partial).

PARTICIPANTS
Participants in the global fairshare policy.

```
$ bgpinfo conf
POLICY_NAME       TYPE
low               fairshare
admin             fairshare

$ bgpinfo conf -l
POLICY NAME: low
-- A global fairshare policy for low queue.
  TYPE: fairshare
  PARAMETERS:
  SYNC_INTERVAL: 30 seconds
  SYNC_MODE: partial
  PARTICIPANTS: low@c1 low@c2

POLICY NAME: admin
-- A global fairshare policy for admin queue.
  TYPE: fairshare
  PARAMETERS:
  SYNC_INTERVAL: 30 seconds
  SYNC_MODE: all
  PARTICIPANTS: admin@c1 admin@c2

$ bgpinfo conf -l admin
POLICY NAME: admin
-- A global fairshare policy for admin queue.
  TYPE: fairshare
  PARAMETERS:
  SYNC_INTERVAL: 30 seconds
  SYNC_MODE: all
  PARTICIPANTS: admin@c1 admin@c2
```

fsload [-l [-c cluster_name]] [global_policy_name ...]
Shows the aggregated fairshare load for global fairshare policies.

By default, displays recursively the summed fairshare load information for the global share account tree for all global fairshare policies. Use the **fsload -l** option to display fairshare load information for each participating queue. Only share accounts whose fairshare load entries are not all zero are shown.

The POLICY_NAME is the name of the global fairshare policy. The following information is displayed for fairshare load:

SHARE_INFO_FOR
The sum of the fairshare load information from all participating queues, users, or user groups in the global fairshare policy.

USER/GROUP
Names of users or user groups who have access to the global fairshare policy.

STARTED
Number of job slots that are used by running or suspended jobs from users or user groups that participate in the global fairshare policy.

RESERVED
Number of job slots that are reserved by jobs from users or user groups that participate in the global fairshare policy.

CPU_TIME
Cumulative CPU time that is used by jobs from users or user groups that participate in the global fairshare policy.

RUN_TIME
Total run time of all participants in the global fairshare policy.

HIST_RUN_TIME
Total historical run time of all participants in the global fairshare policy.

ADJUST
Total dynamic priority calculation adjustment for all participants in the global fairshare policy.

-l

In addition to default output, shows fairshare load information for each participant.

The PROVIDER and SHARE_INFO_FOR show fairshare load information that is provided by a specific participant:

USER/GROUP
Name of users or user groups who have access to the global fairshare policy.

STARTED
Number of job slots that are used by running or suspended jobs from users or user groups for the specific participant.

RESERVED
Number of job slots that are reserved by jobs from users or user groups for the specific participant.

CPU_TIME
Cumulative CPU time that is used by jobs from users or user groups for the specific participant.

RUN_TIME
Run time for the specific participant.

ADJUST
Dynamic priority calculation adjustment for the specific participant.

-c cluster_name

Filter for the `-l` option. It shows fairshare load for the specified participating cluster. Only one cluster name can be specified with the `-c` option.

```
$ bgpinfo fsload -l -c c1 admin
POLICY_NAME: admin

  SHARE_INFO_FOR: /
  USER/GROUP   STARTED  RESERVED  CPU_TIME  RUN_TIME  HIST_RUN_TIME   ADJUST
  ugroup1           27         0       2.9    262464              0    0.000

    PROVIDER: admin@c1
    USER/GROUP   STARTED  RESERVED  CPU_TIME  RUN_TIME  HIST_RUN_TIME   ADJUST
    ugroup1           18         0       2.9     92129              0    0.000

  SHARE_INFO_FOR: /ugroup1/
  USER/GROUP   STARTED  RESERVED  CPU_TIME  RUN_TIME  HIST_RUN_TIME   ADJUST
  user1              9         0       1.0     87490              0     0.000
  user2              9         0       0.0     87488              0     0.000
  user3              9         0       1.0     87486              0     0.000

    PROVIDER: admin@c1
    USER/GROUP   STARTED  RESERVED  CPU_TIME  RUN_TIME  HIST_RUN_TIME   ADJUST
    user1              6         0       1.0     30711              0    0.000
    user2              6         0       0.9     30709              0    0.000
    user3              6         0       1.0     30709              0    0.000
```

status [-l [global_policy_name ...]]

Shows the status of the **gpolicyd** daemon and global fairshare policies. By default, the following information is shown:

GPOLICYD CLUSTER
The cluster where **gpolicyd** is running.

GPOLICYD HOST
The host where **gpolicyd** is running.

-l

In addition to the default output for daemon **gpolicyd** status, shows the status of specified global fairshare policies. By default, shows status of all global fairshare policies.

GLOBAL FAIRSHARE POLICY shows the status of a global fairshare policy:

POLICY NAME
Name of the global fairshare policy.

PARTICIPANTS
Name of participants of the global fairshare policy.

STATUS
Status of the participant.

- `Disconnected` - The cluster of the participant is disconnected from **gpolicyd**
- `Rejected` - Registration rejected. The participant does not exist or the participant is not a fairshare queue.
- ok - The cluster for the participant is connected to **gpolicyd** and the queue for the participant is successfully registered with **gpolicyd**.

```
$ bgpinfo status
GPOLICYD HOST: userA
GPOLICYD CLUSTER: c1

$ bgpinfo status -l
GPOLICYD HOST: userA
GPOLICYD CLUSTER: c1

GLOBAL FAIRSHARE POLICY

POLICY NAME: low
  PARTICIPANTS         STATUS
  low@c1               ok
  low@c2               rejected

POLICY NAME: admin
  PARTICIPANTS         STATUS
  admin@c1             ok
  admin@c2             ok
```

Chapter 14. bhist

Displays historical information about jobs

Synopsis

`bhist` [`-l` [`-aff`] [`-gpu` [`-hostfile`]] [`-a`] [`-b`] [`-d`] [`-data`] [`-e`] [`-p`] [`-r`] [`-s`] [`-w`] [`-UF`] [`-cname`] [`-app` *application_profile_name*] [`-C` *start_time*,*end_time*] [`-D` *start_time*,*end_time*] [`-f` *logfile_name* | `-f -` | `-n` *number_logfiles* | `-n` *number_blocks* | `-n` *min_logfile*, *max_logfile* | `-n 0`] [`-S` *start_time*,*end_time*] [`-J` *job_name*] [`-Jd` "*job_description*"] [`-Lp` *ls_project_name*] [`-m "`*host_name* ..."] [`-N` *host_name* | `-N` *host_model* | `-N` *CPU_factor*] [`-P` *project_name*] [`-q` *queue_name*] [`-u` *user_name* | `-u all` | `-G` *user_group*] [*job_ID* ... | "*job_ID*[*index*]" ...]

`bhist -t` [`-cname`] [`-f` *logfile_name*] [`-T` *start_time*,*end_time*]

`bhist` [`-h` | `-V`]

Description

By default, displays information about your own pending, running, and suspended jobs. Groups information by job. CPU time is not normalized. The **bhist** command searches the event log file that is used by the LSF system: `$LSB_SHAREDIR/`*`cluster_name`*`/logdir/lsb.events`. The **bhist** command also displays events that occurred in the past week. Set the environment variable LSB_BHIST_HOURS to an alternative number of hours (works only with the `-t` option.)

Options

-a
Displays information about both finished and unfinished jobs.

This option overrides `-d`, `-p`, `-s`, and `-r`.

-aff
Displays historical job information about jobs with CPU and memory affinity resource requirement for each task in the job. If the job is pending, the requested affinity resources are displayed. For running jobs, the effective and combined affinity resource allocation is also displayed, along with a table headed `AFFINITY`. The table shows detailed memory and CPU binding information for each task, one line for each allocated processor unit. For finished jobs (`EXIT` or `DONE` state), the affinity requirements for the job, and the effective and combined affinity resource requirement details are displayed.

Use this option only with the `-l` option.

-b
Brief format.

-cname
In IBM Spectrum LSF Advanced Edition, includes the cluster name for execution cluster hosts and host groups in output.

-d
Displays information about finished jobs.

-data
Displays historical information for jobs with data requirements (for example, jobs that are submitted with `-data`). The **bhist -data** option acts as a filter to show only jobs with data requirements.

```
bhist -data
Summary of time in seconds spent in various states:
JOBID   USER    JOB_NAME   PEND    PSUSP   RUN   USUSP   SSUSP   UNKWN   TOTAL
1962    user1 *1000000  410650  0       0     0       0       0       410650
```

-e
Displays information about exited jobs.

-hostfile
If a job was submitted with the **bsub -hostfile** option or modified with the **bmod -hostfile** option to point to a user-specified host file, the **bhist -l -hostfile** option, shows the user-specified host file path. The **-hostfile** option also shows the contents of the host file.

-gpu

bhist -l -gpu shows the following information on GPU job allocation after the job finishes:

Use this option only with the `-l` option.

Host Name
The name of the host.

GPU IDs on the host
Each GPU is shown as a separate line.

TASK and ID
List of job tasks and IDs using the GPU (separated by comma if used by multiple tasks)

MODEL
Contains the GPU brand name and model type name.

MTOTAL
The total GPU memory size.

GPU Compute Capability

MRSV
GPU memory reserved by the job

SOCKET
socket ID of the GPU located at

NVLINK
Indicates if the GPU has NVLink connections with other GPUs allocated for the job (ranked by GPU ID and including itself). The connection flag of each GPU is a character separated by "/" with the next GPU:

A "Y" indicates there is a direct NVLINK connection between two GPUs.

An "N" shows there is no direct NVLINK connection with that GPU.

A "-" shows the GPU is itself.

If the job exited abnormally due to a GPU-related error or warning, the error or warning message displays. If LSF could not get GPU usage information from DCGM, a hyphen (-) displays.

-l
Long format.

If the job was submitted with the **bsub -K** option, the `-l` option displays `Synchronous execution`.

If you submitted a job by using the OR (`||`) expression to specify alternative resources, this option displays the successful `Execution rusage` string with which the job ran.

If you submitted a job with multiple resource requirement strings by using the **bsub -R** option for the `order`, `same`, `rusage`, and `select` sections, the **bhist -l** command displays a single, merged resource requirement string for those sections, as if they were submitted by using a single `-R` option.

Jobs submitted with an **esub** (or **epsub**) using **bsub -a** (or modified using **bmod -a**), will show the latest **esubs** used for execution in **bhist -l** output, first with the default and then user **esubs**. If a user-specified **esub** script is the same as the default **esub** script, the duplicate **esubs** will show as one entry. If a job is submitted with an **esub** containing parameters, the **esub** and its parameters will be shown in **bhist -l** as well, and the format of the **esub** is the same as that specified in the job submission.

The long format includes the following information:

- Job exit codes

- Exit reasons for terminated jobs
- Job exceptions (for example, if job run time exceeds the runtime estimate, a job exception of `runtime_est_exceeded` is displayed).
- Resizable job information
- SSH X11 forwarding information (`-XF`)
- Specified and execution current working directory (CWD). The full path is shown, including directory pattern values.
- Detailed information about jobs with data requirements. The heading `DATA REQUIREMENTS` is displayed followed by a list of the files or tags that are requested by the job, and any modifications made to the data requirements.
- Job kill reason
- Job-level pending time limits and eligible pending time limits.
- The amount of time that the job spent in an eligible and ineligible pending state after the job started, if the **TRACK_ELIGIBLE_PENDINFO** parameter in the `lsb.params` file is set to Y or y.
- Changes to pending jobs as a result of the following **bmod** command options:
  - Absolute priority scheduling (`-aps` | `-apsn`)
  - Autoresizable job attribute (`-ar` | `-arn`)
  - Current working directory (`-cwd`)
  - Data requirements (`-data` | `-datan`)
  - Post-execution command (`-Ep` | `-Epn`)
  - Job description (`-Jd` | `-Jdn`)
  - Checkpoint options (`-k` | `-kn`)
  - Migration threshold (`-mig` | `-mign`)
  - Job resize notification command (`-rnc` | `-rncn`)
  - User limits (`-ul` | `-uln`)
  - Runtime estimate (`-We` | `-Wen`)

The **bhist -l** command displays the effective GPU requirements string for a GPU allocation.

```
bhist -l
Job <204>, User <user1>, Project <default>, Command <blaunch sleep 60>
Wed Jul 12 22:40:54: Submitted from host <hosta>, to Queue <normal>, CWD </
                     scratch/user1>, 8 Task(s),Requested Resources <span[ptile=4]
.....................rusage[ngpus_physical=4]>,Specified Hosts <hostb>,
                     <hosta!>, Requested GPU <num=4:mode=shared:j_exclusive=yes>;
Wed Jul 12 22:40:55: Dispatched 8 Task(s) on Host(s) <hosta> <hosta> <h
                     hosta> <hosta> <hostb> <hostb> <hostb>
                     <hostb>, Allocated 8 Slot(s) on Host(s) <hosta> <h
                     hosta> <hosta> <hosta> <hostb> <hostb>
                     <hostb> <hostb>, Effective RES_REQ <select[type ==
                      any] order[r15s:pg] rusage[ngpus_physical=4.00] span[ptil
                     e=4] >;
Wed Jul 12 22:40:56: Starting (Pid 116194);
Wed Jul 12 22:40:56: External Message "hostb:gpus=0,3,1,2;hosta:gpus=0,1,2,3;
                     EFFECTIVE GPU REQ: num=4:mode=shared:mps=no:j_exclusive=yes;"
                     was posted from "user1" to message box 0;
```

-p

Displays information about pending jobs.

-r

Displays information about running jobs.

-s

Displays information about suspended jobs.

-t
Displays job events chronologically, including energy aware scheduling events JOB_PROV_HOST and HOST_POWER_STATUS.

By default, displays only records from the last week. For different time periods, use the -t option with the -T option.

Use **LSB_BHIST_HOURS** with -t to display job events before the current time for a longer or shorter period than the default one week.

-w
Wide format. Displays the information in a wide format.

-UF
Displays unformatted job detail information.

This option makes it easy to write scripts for parsing keywords on **bhist**. The results of this option have no wide control for the output. Each line starts from the beginning of the line. The resource usage message lines that end without any separator have a semicolon added to separate their different parts. The first line and all lines that start with the time stamp are displayed unformatted in a single line. The output has no line length and format control.

-app *application_profile_name*
Displays information about jobs that are submitted to the specified application profile.

-C *start_time,end_time*
Displays jobs that completed or exited during the specified time interval. Specify the times in the format yyyy/mm/dd/HH:MM. Do not specify spaces in the time interval string. This option overrides the -r, -s, -p, and -a options.

For more information about the time syntax, see Time interval format.

-D *start_time,end_time*
Displays jobs that are dispatched during the specified time interval. Specify the times in the format yyyy/mm/dd/HH:MM. Do not specify spaces in the time interval string.

Must be used with the -a option since it finds results only in running jobs.

For more information about the time syntax, see Time interval format.

-f *logfile_name* | -f -
Searches the specified event log, which is useful for offline analysis. Specify either an absolute or a relative path.

The specified file path can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

Specify the -f - option to force the **bhist** command to use the lsb.events log file. If you are using IBM Spectrum LSF Explorer (LSF Explorer) to load event log records, the -f - option (or any -f argument that specifies a log file) forces the **bhist** command to bypass LSF Explorer.

Note: The **bhist** -cname, -t, and -T options always bypass LSF Explorer and instead get the information from the lsb.events file.

For more details, refer to **LSF_QUERY_ES_SERVERS** and **LSF_QUERY_ES_FUNCTIONS** in the *IBM Spectrum LSF Configuration Reference*.

-G *user_group*
Displays jobs that are associated with a user group that is submitted with the **bsub -G** command for the specified user group. The -G option does not display jobs from subgroups within the specified user group.

The -G option cannot be used together with the -u option. You can specify only a user group name. The keyword all is not supported for the -G option.

-J *job_name*
Displays the jobs that have the specified job name.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-Jd "*job_description*"
Displays the jobs that have the specified job description.

The job description can be up to 4094 characters long. Job descriptions are not unique.

The wildcard character (*) can be used anywhere within a job description.

-Lp *ls_project_name*
Displays information about jobs that belong to the specified LSF License Scheduler project.

-m "*host_name*..."
Displays jobs that are dispatched to the specified host.

-n *number_logfiles* | -n *min_logfile*, *max_logfile* | -n 0
Searches the specified number of event logs, starting with the current event log and working through the most recent logs in consecutive order. Specify 0 to specify all the event log files in $(LSB_SHAREDIR)/*cluster_name*/logdir, up to a maximum of 100 files.

If you delete a file, you break the consecutive numbering, and older files are inaccessible to the **bhist** command. For example, if you specify 3, LSF searches the `lsb.events`, `lsb.events.1`, and `lsb.events.2` files. If you specify 4, LSF searches the `lsb.events`, `lsb.events.1`, `lsb.events.2`, and `lsb.events.3` files. However, if the `lsb.events.2`file is missing, both searches include only the `lsb.events` and `lsb.events.1` files.

-n *number_blocks* (LSF Explorer only)
If you are using LSF Explorer to load event log records, use the `-n` option to control how many job records (number of block records) that LSF Explorer returns. The block size is configured in LSF Explorer. For more details, refer to **LSF_QUERY_ES_SERVERS** and **LSF_QUERY_ES_FUNCTIONS** in the *IBM Spectrum LSF Configuration Reference*.

-N *host_name* | -N *host_model* | -N *cpu_factor*
Normalizes CPU time by the specified CPU factor, or by the CPU factor of the specified host or host model.

If you use the **bhist** command directly on an event log, you must specify a CPU factor.

Use the **lsinfo** command to get host model and CPU factor information.

-P *project_name*
Displays information about jobs that belong to the specified project.

-q *queue_name*
Displays information about jobs that are submitted to the specified queue.

-S *start_time*,*end_time*
Displays information about jobs that are submitted during the specified time interval. Specify the times in the format `yyyy/mm/dd/HH:MM`. Do not specify spaces in the time interval string.

Must be used with the `-a` option since it finds results only in running jobs.

For more information about the time syntax, see Time interval format.

-T *start_time*,*end_time*
Used together with the `-t` option.

Displays information about job events within the specified time interval. Specify the times in the format `yyyy/mm/dd/HH:MM`. Do not specify spaces in the time interval string.

For more information about the time syntax, see Time interval format.

-u *user_name* | -u all
Displays information about jobs that are submitted by the specified user, or by all users if the keyword `all` is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash in a Windows command (*DOMAIN_NAME**user_name*), or a double backslash in a UNIX command (*DOMAIN_NAME**user_name*).

***job_ID* | "*job_ID*[*index*]" ...**
Searches all event log files and displays only information about the specified jobs. If you specify a job array, displays all elements chronologically.

You specify job ID when you know exactly which jobs you want, so do not specify any other options that control job selection (`-a, -d, -e, -p, -r, -s, -D, -S, -T, -app, -G, -J, -Jd, -Lp, -M, -q, -u`). If you specify an illogical combination of selection criteria, the system does not return any matching jobs.

In LSF multicluster capability job forwarding mode, you can use the local job ID and cluster name to retrieve the job details from the remote cluster. Use the following query syntax:

```
bhist submission_job_id@submission_cluster_name
```

For job arrays, use the following query syntax:

```
bhist "submission_job_id[index]"@submission_cluster_name
```

The advantage of using **src_job_id@src_cluster_name** instead of **bhist -l job_id** is that you can use **src_job_id@src_cluster_name** as an alias to query a local job in the execution cluster without knowing the local job ID in the execution cluster. The **bhist** command output is identical no matter which job ID you use (local job ID or `src_job_id@src_cluster_name`).

You can use the **bhist 0** option to find all historical jobs in your local cluster, but the **bhist 0@submission_cluster_name** option is not supported.

-h
Prints command usage to `stderr` and exits.

-V
Prints release version to `stderr` and exits.

Output: Default format

Memory Usage
Displays peak memory usage and average memory usage.

```
MEMORY USAGE:
MAX MEM: 11 Mbytes; AVG MEM:6 Mbytes
```

If consumed memory is larger or smaller than current rusage, you can adjust resource requirements next time for the same job submission.

Time Summary
Statistics of the amount of time that a job spent in various states.

PEND
The total waiting time, excluding user suspended time before the job is dispatched.

PSUSP
The total user suspended time of a pending job.

RUN
The total run time of the job.

USUSP
The total user suspended time after the job is dispatched.

SSUSP
The total system suspended time after the job is dispatched.

UNKWN
The total unknown time of the job (job status becomes unknown if the **sbatchd** daemon on the execution host is temporarily unreachable).

TOTAL
The total time that the job spent in all states. For a finished job, it is the turnaround time, which is the time interval from job submission to job completion.

Output: Long format (-l)

The `-l` option displays a long format listing with the following extra fields:

Project
The project the job was submitted from.

Application Profile
The application profile the job was submitted to.

Command
The job command.

Detailed history includes job group modification, the date, and time the job was forwarded and the name of the cluster to which the job was forwarded.

The displayed job command can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

Initial checkpoint period
The initial checkpoint period that is specified at the job level with the **bsub -k** option, or in an application profile with the **CHKPNT_INITPERIOD** parameter in the `lsb.applications` file.

Checkpoint period
The checkpoint period that is specified at the job level with the **bsub -k** option, in the queue with the **CHKPNT** parameter in the `lsb.queues` file, or in an application profile with the **CHKPNT_PERIOD** parameter in the `lsb.applications` file.

Checkpoint directory
The checkpoint directory that is specified at the job level with the **bsub -k** option, in the queue with the **CHKPNT** parameter in the `lsb.queues` file, or in an application profile with the **CHKPNT_DIR** parameter in the `lsb.applications` file.

Migration threshold
The migration threshold that is specified at the job level, with the **bsub -mig** option.

Requested Resources
Shows all the resource requirement strings that you specified in the **bsub** command.

Execution CWD
The actual current working directory (CWD) that is used when job runs.

Host file
The path to a user-specified host file that is used when you submitted or modified the job.

Execution Rusage
Shown if the combined RES_REQ has an `rusage` OR || construct. The chosen alternative is denoted here.

Effective RES_REQ
Displays a job's resource requirement as seen by the scheduler after it resolves any OR constructs.

Resizable job information

- For `JOB_NEW` events, the **bhist** command displays the autoresizable attribute and resize notification command in the submission line.
- For `JOB_MODIFY2` events from the **bmod** command, the **bhist** command displays the autoresizable attribute and resize notification command in the submission line.

- **bmod -arn** *jobID*

```
Parameters of Job are changed: Autoresizable attribute is removed;
```

- **bmod -ar** *jobID*

```
Parameters of Job are changed: Job changes to autoresizable;
```

- **bmod -rnc** *resize_notification_cmd jobID*

```
Parameters of Job are changed: Resize notification command changes to:
<resize_notification_cmd>;
```

- **bmod -rncn** *jobID*

```
Parameters of Job are changed: Resize notification command is removed;
```

- For the JOB_RESIZE_NOTIFY_START event, the **bhist** command displays the following message:

```
Added <num_tasks> tasks on host <host_list>, <num_slots> additional slots allocated on
<host_list>
```

- For the JOB_RESIZE_NOTIFY_ACCEPT event, the **bhist** command displays the following messages:
  - If the notification command is configured and the **sbatchd** daemon successfully initializes the notification command, the **bhist** command displays the following message:

```
Resize notification accepted. Notification command initialized (Command
PID: 123456)
```

  - If a notification command is not defined, the **bhist** displays the following message:

```
Resize notification accepted
```

  - If the **sbatchd** daemon reports failure for whatever reason, the **bhist** displays the following message:

```
Resize notification failed
```

- For the JOB_RESIZE_NOTIFY_DONE event, the **bhist** command displays the following messages:
  - If status is 0, `Resize notification command completed`
  - If status is 1, `Resize notification command failed`
- For JOB_RESIZE_RELEASE event, the **bhist** command displays the following information:

```
Release allocation on <num_hosts> Hosts/Processors <host_list> by user or
administrator <user_name>
Resize notification accepted;
```

  For the **bmod -rncn** option, the **bhist** command displays the following message:

```
Resize notification command disabled
```

- For JOB_RESIZE_CANCEL event, the **bhist** command displays the following message:

```
Cancel pending allocation request
```

Synchronous execution
Job was submitted with the **-K** option. LSF submits the job and waits for the job to complete.

Terminated jobs: exit reasons
For terminated jobs, displays exit reasons.

Interactive jobs
For interactive jobs, the **bhist -l** command does not display information about the execution home directory, current working directory, or running PID for the job.

Dispatched <number> Task(s) on Host(s)
The number of tasks in the job and the hosts to which those tasks were sent for processing. Displayed if the **LSB_ENABLE_HPC_ALLOCATION=Y** parameter is set in the `lsf.conf` file.

Allocated <number> Slot(s) on Host(s)
The number of slots that were allocated to the job based on the number of tasks, and the hosts on which the slots are allocated. Displayed if the **LSB_ENABLE_HPC_ALLOCATION=Y** parameter is set in the `lsf.conf` file.

Requested Network and PE Network ID
Network resource requirements for IBM Parallel Edition (PE) jobs that are submitted with the `bsub -network` option, or if the **NETWORK_REQ** parameter is specified in a queue (defined in the `lsb.queues` file) or an application profile (defined in the `lsb.applications` file).

```
bhist -l 749
Job <749>, User <user1>;, Project <default>, Command <my_pe_job>

Mon Jun  4 04:36:12: Submitted from host <hostB>, to Queue <
                     priority>, CWD <$HOME>, 2 Tasks, Requested
                     Network <type=sn_all:protocol=mpi:mode=US:usage=
                     shared:instance=1>;
Mon Jun  4 04:36:15: Dispatched 2 Task(s) on Host(s) <hostB>,
                     Allocated <1> Slot(s) on Host(s) <hostB>;
                     Effective RES_REQ <select[type == local] rusage
                     [nt1=1.00] >, PE Network ID <1111111>  <2222222>
                     used <1> window(s)per network per task;
Mon Jun  4 04:36:17: Starting (Pid 21006);
```

DATA REQUIREMENTS
The `-l -data` option displays a list of requested files or tags for jobs with data requirements and any modifications to data requirements.

Output: Affinity resource requirements information (-l -aff)

Use the `-l -aff` option to display historical job information about CPU and memory affinity resource requirements for job tasks. A table with the heading AFFINITY is displayed containing the detailed affinity information for each task, one line for each allocated processor unit. CPU binding and memory binding information are shown in separate columns in the display.

HOST
The host the task is running on.

TYPE
Requested processor unit type for CPU binding. One of `numa`, `socket`, `core`, or `thread`.

LEVEL
Requested processor unit binding level for CPU binding. One of `numa`, `socket`, `core`, or `thread`. If no CPU binding level is requested, a dash (`-`) is displayed.

EXCL
Requested processor unit binding level for exclusive CPU binding. One of `numa`, `socket`, or `core`. If no exclusive binding level is requested, a dash (`-`) is displayed.

IDS
List of physical or logical IDs of the CPU allocation for the task.

The list consists of a set of paths, represented as a sequence of integers separated by slash characters (`/`), through the topology tree of the host. Each path identifies a unique processing unit that is allocated to the task. For example, a string of the form `3/0/5/12` represents an allocation to thread 12 in core 5 of socket 0 in NUMA node 3. A string of the form `2/1/4`represents an allocation to core 4 of socket 1 in NUMA node 2. The integers correspond to the node ID numbers displayed in the topology tree from the **bhosts -aff** command.

POL
Requested memory binding policy. Either `local` or `pref`. If no memory binding is requested, a dash (-) is displayed.

NUMA
ID of the NUMA node that the task memory is bound to. If no memory binding is requested, a dash (-) is displayed.

SIZE
Amount of memory that is allocated for the task on the NUMA node.

For example, the following job starts six tasks with the following affinity resource requirements:

```
bsub -n 6 -R"span[hosts=1] rusage[mem=100]affinity[core(1,same=socket,
exclusive=(socket,injob)):cpubind=socket:membind=localonly:distribute=pack]" myjob
Job <6> is submitted to default queue <normal>.
```

```
bhist -l -aff 6

Job <6>, User <user1>, Project <default>, Command <myjob>
Thu Feb 14 14:13:46: Submitted from host <hostA>, to Queue <normal>, CWD <$HO
                     ME>, 6 Task(s), Requested Resources <span[hos
                     ts=1] rusage[mem=100]affinity[core(1,same=socket,exclusive
                     =(socket,injob)):cpubind=socket:membind=localonly:distribu
                     te=pack]>;
Thu Feb 14 14:15:07: Dispatched 6 Task(s) on Host(s) <hostA> <hostA> <hostA>
                     <hostA> <hostA> <hostA>; Allocated <6> Slot(s) on Host(s)
                     <hostA> <hostA> <hostA> <hostA> <hostA> <hostA>;
                     Effective RES_REQ <select[type == local] order[r15s:pg]
                     rusage[mem=100.00] span[hosts=1] affinity [core(1,same=
                     socket,exclusive=(socket,injob))*1:cpubind=socket:membind=
                     localonly:distribute=pack] >;

AFFINITY:
                    CPU BINDING                          MEMORY BINDING
                    -----------------------              --------------------
HOST                TYPE   LEVEL  EXCL   IDS            POL   NUMA SIZE
hostA               core   socket socket /0/0/0         local 0    16.7MB
hostA               core   socket socket /0/1/0         local 0    16.7MB
hostA               core   socket socket /0/2/0         local 0    16.7MB
hostA               core   socket socket /0/3/0         local 0    16.7MB
hostA               core   socket socket /0/4/0         local 0    16.7MB
hostA               core   socket socket /0/5/0         local 0    16.7MB

Thu Feb 14 14:15:07: Starting (Pid 3630709);
Thu Feb 14 14:15:07: Running with execution home </home/jsmith>, Execution CWD
                     </home/jsmith>, Execution Pid <3630709>;
Thu Feb 14 14:16:47: Done successfully. The CPU time used is 0.0 seconds;
Thu Feb 14 14:16:47: Post job process done successfully;

MEMORY USAGE:
MAX MEM: 2 Mbytes;  AVG MEM: 2 Mbytes

Summary of time in seconds spent in various states by  Thu Feb 14 14:16:47
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  81       0        100      0        0        0        181
```

Output: Data requirements information (-l -data)

Use the `-l -data` option to display detailed information about jobs with data requirements. The heading `DATA REQUIREMENTS` is displayed followed by a list of the files or tags that are requested by the job, and any modifications made to the data requirements.

```
bhist -data -l 84046
Job <84046>, User <user1>, Project <default>, Command <bstage out -src /home/
                     user1/data2; bstage out -src /home/user1/data2 -dst /tmp;
                     datajob1.sh>
Mon Aug 18 15:06:57: Submitted from host <hostA>, to Queue <normal>, CWD </sc
                     ratch/user1/workspace/simulation/data/>, Data Require
                     ment Requested;

DATA REQUIREMENTS:

 FILE: hostA:/home/user1/data2
```

```
 SIZE: 40 MB
 MODIFIED: Thu Aug 14 17:01:57

 FILE: hostA:/home/user1/data3
 SIZE: 45 MB
 MODIFIED: Fri Aug 15 16:32:45
Mon Aug 18 15:07:07: Dispatched to <hostB>, Effective RES_REQ <select[type ==
                     local] order[r15s:pg] >;
Mon Aug 18 15:07:07: Starting (Pid 16128);
Mon Aug 18 15:07:07: Running with execution home </home/user1>, Execution CWD <
                     /scratch/user1/workspace/simulation/data/>, Execution
                     Pid <16128>;
Mon Aug 18 15:08:47: Done successfully. The CPU time used is 0.4 seconds;
Mon Aug 18 15:08:47: Post job process done successfully;

MEMORY USAGE:
MAX MEM: 3 Mbytes;  AVG MEM: 2 Mbytes

Summary of time in seconds spent in various states by  Mon Aug 18 15:08:47
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  10       0        100      0        0        0        110
```

Output: User-specifed host file (-l -hostfile)

Use the `-l -hostfile` option to display a user-specified host file that was submitted with a job or added to a job with the **bmod** command.

```
bhist -l -hostfile  1976
Job <1976>, User <user1>, Project <default>, Command <my_data_job>
Fri Sep 20 16:31:17: Submitted from host <hostA>, to
           Queue <normal>, CWD <$HOME/source/user1/work>,
           Host file </home/user4/myhostfile>;

Summary of time in seconds spent in various states by  Wed Sep 25 10:50:37
PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
21305      0        0        0        0        0      21305
USER-SPECIFIED HOST FILE:
HOST            SLOTS
host01            3
host02            1
host01            1
host02            2
host03            1
```

Files

Reads the `lsb.events` file.

See also

`lsb.events` file, **bgadd**, **bgdel**, **bjgroup**, **bsub**, **bjobs**, **lsinfo** commands

Time interval format

You use the time interval to define a start and end time for collecting the data to be retrieved and displayed. While you can specify both a start and an end time, you can also let one of the values default. You can specify either of the times as an absolute time, by specifying the date or time, or you can specify them relative to the current time.

Specify the time interval:

start_time,*end_time*|*start_time*,|,*end_time*|*start_time*

Specify *start_time* or *end_time* in the following format:

[*year*/][*month*/][*day*][/*hour*:*minute*|/*hour*:]|.|.-*relative_int*

year
Four-digit number that represents the calendar year.

month
Number 1 - 12, where 1 is January and 12 is December.

day
Number 1 - 31, representing the day of the month.

hour
Integer 0 - 23, representing the hour of the day on a 24-hour clock.

minute
Integer 0 - 59, representing the minute of the hour.

. (period)
Represents the current `month/day/hour:minute`.

.-*relative_int*
Number, 1 - 31, specifying a relative start or end time before now.

start_time,end_time
Specifies both the start and end times of the interval.

start_time,
Specifies a start time, and lets the end time default to now.

,end_time
Specifies to start with the first logged occurrence, and end at the time specified.

start_time
Starts at the beginning of the most specific time period that is specified, and ends at the maximum value of the time period specified. For example, `2/` specifies the month of February — start February 1 at 00:00 AM and end at the last possible minute in February (28 February at midnight).

Absolute time examples

Assume that the current time is `May 9 17:06 2008`:

`1,8` = `May 1 00:00 2008` to `May 8 23:59 2008`

`,4` = the time of the first occurrence to `May 4 23:59 2008`

`6` = `May 6 00:00 2008` to `May 6 23:59 2008`

`2/` = `Feb 1 00:00 2008` to `Feb 28 23:59 2008`

`/12:` = `May 9 12:00 2008` to `May 9 12:59 2008`

`2/1` = `Feb 1 00:00 2008` to `Feb 1 23:59 2008`

`2/1,` = `Feb 1 00:00` to the current time

`,.` = the time of the first occurrence to the current time

`,2/10:` = the time of the first occurrence to `May 2 10:59 2008`

`2001/12/31,2008/5/1` = from `Dec 31 2001 00:00:00` to `May 1 2008 23:59:59`

Relative time examples

`.-9,` = `April 30 17:06 2008` to the current time

`,.-2/` = the time of the first occurrence to `Mar 7 17:06 2008`

`.-9,.-2` = nine days ago to two days ago (`April 30 17:06 2008` to `May 7 17:06 2008`)

Chapter 15. bhosts

Displays hosts and their static and dynamic resources

Synopsis

`bhosts` [`-w` | `-l` | `-e` | `-o "`*field_name*[`:`[`-`][*output_width*]] ... [`delimiter='`*character*`']"` [`-json`]] [`-a`] [`-alloc`] [`-cname`] [`-x`] [`-X`] [`-R "`*res_req*`"`] [*host_name* ... | *host_group* ... | *compute_unit* ...]

`bhosts` [`-w` | `-e` | `-o "`*field_name*[`:`[`-`][*output_width*]] ... [`delimiter='`*character*`']"` [`-json`]] [`-a`] [`-alloc`] [`-cname`] [`-noheader`] [`-x`] [`-X`] [`-R "`*res_req*`"`][*host_name* ... | *host_group* ... | *compute_unit* ...]

`bhosts -s` [*resource_name* ...] [`-a`] [`-cname`] [`-e`] [`-noheader`]

`bhosts` [`-aff`] [`-l`] [*host_name* ... | *host_group* ... | *compute_unit* ...] | [*cluster_name*]

`bhosts` [`-w` | `-l` | `-e` | `-o "`*field_name*[`:`[`-`][*output_width*]] ... [`delimiter='`*character*`']"`] [`-a`] [`-alloc`] [`-cname`] [`-gpu`] [`-x`] [`-X`] [`-R "`*res_req*`"`] [`-rc`] [*host_name* ... | cluster_name]

`bhosts` [`-w`] [`-rconly`]

`bhosts` [`-h` | `-V`]

Description

By default, returns the following information about all hosts: Host name, host status, job state statistics, and job slot limits.

The **bhosts** command displays output for condensed host groups and compute units. These host groups and compute units are defined by `CONDENSE` in the `HostGroup` and `ComputeUnit` sections of the `lsb.hosts` file. Condensed host groups and compute units are displayed as a single entry with the name as defined by `GROUP_NAME` or `NAME` in the `lsb.hosts` file.

When EGO adds more resources to a running resizable job, the **bhosts** command displays the added resources. When EGO removes resources from a running resizable job, the **bhosts** command displays the updated resources.

The `-l` and `-X` options display uncondensed output.

The `-s` option displays information about the numeric shared resources and their associated hosts.

With LSF multicluster capability, displays the information about hosts available to the local cluster. Use the `-e` option to see information about exported hosts.

Options

-a
Shows information about all hosts, including hosts relinquished to a resource provider (such as EGO or OpenStack) through LSF resource connector. Default output includes only standard LSF hosts.

-aff
Displays host topology information for CPU and memory affinity scheduling.

-alloc
Shows counters for slots in `RUN`, `SSUSP`, `USUSP`, and `RSV`. The slot allocation is different depending on whether the job is an exclusive job or not.

-cname
In LSF Advanced Edition, includes the cluster name for execution cluster hosts in output. The output that is displayed is sorted by cluster and then by host name.

-e
LSF multicluster capability only. Displays information about resources that were exported to another cluster.

-gpu [-l]
Displays GPU information on the host.

The `-l` option shows more detailed information about the GPUs.

-json

Displays the customized output in JSON format.

When specified, **bhosts -o** displays the customized output in the JSON format.

This option applies only to output for the **bhosts -o** command for customized output. This option has no effect when used with **bhosts** without the `-o` option and the **LSB_BHOSTS_FORMAT** environment variable and parameter are not defined.

-l
Displays host information in a long multi-line format. In adslotsdition to the default fields, displays information about the CPU factor, the current load, and the load thresholds. Also displays the value of for each host. The `slots` value is the greatest number of unused slots on a host.

The **bhosts -l** option also displays information about the dispatch windows.

When PowerPolicy is enabled in the `lsb.threshold` file, the **bhosts -l** command also displays host power states. Final power states are on or `suspend`. Intermediate power states are `restarting`, `resuming`, and `suspending`. The final power state under administrator control is `closed_Power`. The final power state under policy control is ok_Power. If the host status becomes unknown (power operation due to failure), the power state is shown as a dash (`-`).

If you specified an administrator comment with the `-C` option of the host control commands (**badmin hclose -C** or **badmin hopen -C**), the `-l` option displays the comment text.

If enhanced energy accounting using Elasticsearch has been enabled (with **LSF_ENABLE_BEAT_SERVICE** in `lsf.conf`), output will show the **Current Power** usage in watts, and total **Energy Consumed** in Joule and kWh.

-noheader

Removes the column headings from the output.

When specified, **bhosts** displays the values of the fields without displaying the names of the fields. This option is useful for script parsing, when column headings are not necessary.

This option applies to output for the **bhosts** command with no options, and to output for all **bhosts** options with output that uses column headings, including the following options: `-a`, `-alloc`, `-cname`, `-e`, `-o`, `-R`, `-s`, `-w`, `-x`, `-X`.

This option does not apply to output for **bhosts** options that do not use column headings, including the following options: `-aff`, `-json`, `-l`.

-o

Sets the customized output format.

- Specify which **bhosts** fields (or aliases instead of the full field names), in which order, and with what width to display.
- Specify only the **bhosts** field name or alias to set its output to unlimited width and left justification.
- Specify the colon (`:`) without a width to set the output width to the recommended width for that field.
- Specify the colon (`:`) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **bhosts** truncates the ending characters.
- Specify a hyphen (`-`) to set right justification when **bhosts** displays the output for the specific field. If not specified, the default is to set left justification when **bhosts** displays the output for a field.
- Use `delimiter=` to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the output for certain **bhosts** options:

- **LSB_BHOSTS_FORMAT** and **bhosts -o** both apply to output for the **bhosts** command with no options, and for **bhosts** options with output that filter information, including the following options: `-a`, `-alloc`, `-cname`, `-R`, `-x`, `-X`.
- **LSB_BHOSTS_FORMAT** and **bhosts -o** do not apply to output for **bhosts** options that use a modified format, including the following options: `-aff`, `-e`, `-l`, `-s`, `-w`.

The **bhosts -o** option overrides the **LSB_BHOSTS_FORMAT** environment variable, which overrides the **LSB_BHOSTS_FORMAT** setting in `lsf.conf`.

The following are the field names used to specify the **bhosts** fields to display, recommended width, aliases you can use instead of field names, and units of measurement for the displayed field:

Table 1. Output fields for bhosts

Field name	Width	Aliases	Unit
host_name	20	hname	
status	15	stat	
cpuf	10		
jl_u	8	jlu	
max	8		
njobs	8		
run	8		
ssusp	8		
ususp	8		
rsv	8		
dispatch_window	50	dispwin	
ngpus	8	ng	
ngpus_alloc	8	ngu	
ngpus_excl_alloc	8	ngx	
ngpus_shared_alloc	8	ngs	
ngpus_shared_jexcl_alloc	8	ngsjx	
ngpus_excl_avail	8	ngfx	
ngpus_shared_avail	8	ngfs	

Field names and aliases are not case-sensitive. Valid values for the output width are any positive integer 1 - 4096.

For example,

```
bhosts -o "host_name cpuf: jl_u:- max:-6 delimiter='^'"
```

This command displays the following fields:

- HOST_NAME with unlimited width and left justified.
- CPUF with a maximum width of ten characters (which is the recommended width) and left justified.
- JL_U with a maximum width of eight characters (which is the recommended width) and right justified.
- MAX with a maximum width of six characters and right justified.

- The ^ character is displayed between different headers and fields.

-w

Displays host information in wide format. Fields are displayed without truncation.

For condensed host groups and compute units, the `-w` option displays the overall status and the number of hosts with the ok, `unavail`, `unreach`, and `busy` status in the following format:

host_group_status num_ok/num_unavail/num_unreach/num_busy

Where

- *host_group_status* is the overall status of the host group or compute unit. If a single host in the group or unit is ok, the overall status is also ok.
- *num_ok*, *num_unavail*, *num_unreach*, and *num_busy* are the number of hosts that are ok, `unavail`, `unreach`, and `busy`.

For example, if five hosts are ok, two `unavail`, one `unreach`, and three `busy` in a condensed host group `hg1`, the following status is displayed:

```
hg1 ok 5/2/1/3
```

If any hosts in the host group or compute unit are closed, the status for the host group is displayed as `closed`, with no status for the other states:

```
hg1 closed
```

The status of LSF resource connector hosts that are closed because of a resource provider reclaim request is `closed_RC`.

-rc [-l]

Displays the current status of hosts requested from and provisioned by LSF resource connector, as well as a brief history of each provisioned host.

Note: Requires LSF Fix Pack 4.

The `-rc` and `-rconly` options make use of the third-party **mosquitto** message queue application. LSF resource connector publishes additional provider host information that is displayed by these **bhosts** options. The **mosquitto** binary file is included as part of the LSF distribution.

To use the `-rc` option, LSF resource connector must be enabled with the **LSB_RC_EXTERNAL_HOST_FLAG** parameter in the `lsf.conf` file.

If you use the MQTT message broker that is distributed withLSF, you must configure the **LSF_MQ_BROKER_HOSTS** and **MQTT_BROKER_HOST** parameters in the `lsf.conf` file. The **LSF_MQ_BROKER_HOSTS** and **MQTT_BROKER_HOST** parameters must specify the same host name. The **LSF_MQ_BROKER_HOSTS** parameter enables LIM to start the **mosquitto** daemon.

If you use an existing MQTT message broker, you must configure the **MQTT_BROKER_HOST** parameter. You can optionally specify an MQTT broker port with the **MQTT_BROKER_PORT** parameter.

Use the **ps** command to check that the MQTT message broker daemon (**mosquitto**) is installed and running: `ps -ef | grep mosquitto`.

Configure the **EBROKERD_HOST_CLEAN_DELAY** to specify a delay, in minutes, after which the **ebrokerd** daemon removes information about relinquished or reclaimed hosts. This parameter allows the **bhosts -rc** and **bhosts -rconly** commands to get LSF resource connector provider host information for some time after they are deprovisioned.

The following additional columns are shown in the host list:

RC_STATUS

LSF resource connector status.

Preprovision_Started

Resource connector started the preprovisioning script for the new host.

Preprovision_Failed
The preprovisioning script returned an error.

Allocated
The host is ready to join the LSF cluster.

Reclaim_Received
A host reclaim request was received from the provider (for example, for an AWS spot instance).

RelinquishReq_Sent
LSF started to relinquish the host.

Relinquished
LSF finished relinquishing the host.

Deallocated_Sent
LSF sent a return request to the provider.

Postprovision_Started
LSF started the postprovisioning script after the host was returned.

Done
The host life cyle is complete.

PROV_STATUS
Provider status. This status depends the provider. For example, AWS has pending, running, shutting down, terminated, and others. Check documentation for the provider to understand the status that is displayed.

UPDATED_AT
Time stamp of the latest status change.

INSTANCE_ID
ID of the created machine instance. This provides a unique ID for each cloud instance of the LSF resource connector host.

For hosts provisioned by resource connector, these columns show appropriate status values and a time stamp. A dash (-) is displayed in these columns for other hosts in the cluster.

For example,

```
bhosts -rc
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV  RC_STATUS
PROV_STATUS    UPDATED_AT              INSTANCE_ID
ec2-35-160-173-192 ok              -      1      0      0      0      0      0  Allocated
running        2017-04-07T12:28:46CDT  i-0244f608fe7b5e014
lsf1.aws.          closed          -      1      0      0      0      0      0          -
-              -
```

The `-l` option shows more detailed information about provisioned hosts:

```
bhosts -rc -l
HOST  ec2-35-160-173-192.us-west-2.compute.amazonaws.com
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV RC_STATUS      PROV_STATUS
UPDATED_AT             INSTANCE_ID              DISPATCH_WINDOW
ok             60.00     -      1      0      0      0      0      0 Allocated      running
2017-04-07T12:28:46CDT i-0244f608fe7b5e014      -

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem  slots
 Total           1.0   0.0   0.0    1%   0.0    33    0     3 5504M    0M  385M      1
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M      -
```

-rconly
Shows the status of all hosts provisioned by LSF resource connector, no matter if they have joined the cluster or not.

Note: Requires LSF Fix Pack 4.

To use the `-rconly` option, LSF resource connector must be enabled with the **LSB_RC_EXTERNAL_HOST_FLAG** parameter in the `lsf.conf` file. If you use the MQTT message broker that is distributed withLSF, you must configure the **LSF_MQ_BROKER_HOSTS** and **MQTT_BROKER_HOST** parameters in the `lsf.conf` file. The **LSF_MQ_BROKER_HOSTS** and **MQTT_BROKER_HOST** parameters must specify the same host name. The **LSF_MQ_BROKER_HOSTS** parameter enables LIM to start the **mosquitto** daemon.

If you use an existing MQTT message broker, you must configure the **MQTT_BROKER_HOST** parameter. You can optionally specify an MQTT broker port with the **MQTT_BROKER_PORT** parameter.

Use the **ps** command to check that the MQTT message broker daemon (**mosquitto**) is installed and running: `ps -ef | grep mosquitto`.

-x

Display hosts whose job exit rate is high and exceeds the threshold that is configured by the **EXIT_RATE** parameter in the `lsb.hosts` file for longer than the value specified by the **JOB_EXIT_RATE_DURATION** parameter that is configured in the `lsb.params` file. By default, these hosts are closed the next time LSF checks host exceptions and runs **eadmin**.

Use with the `-l` option to show detailed information about host exceptions.

If no hosts exceed the job exit rate, the **bhosts -x** command has the following output:

```
There is no exceptional host found
```

-X

Displays uncondensed output for host groups and compute units.

-R "*res_req*"

Displays only information about hosts that satisfy the resource requirement expression.

Note: Do not specify resource requirements by using the `rusage` keyword to select hosts because the criteria are ignored by LSF.

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

-s [*resource_name* ...]

Specify shared numeric resources only. Displays information about the specified resources. Returns the following information: the resource names, the total and reserved amounts, and the resource locations.

The **bhosts -s** option shows only consumable resources.

When the **LOCAL_TO** parameter is configured for a license feature in the `lsf.licensescheduler` file, the **bhosts -s** command shows different resource information, depending on the cluster locality of the features.

From `clusterA`:

```
bhosts -s
RESOURCE                TOTAL       RESERVED       LOCATION
hspice                  36.0        0.0            host1
```

From `clusterB` in `siteB`:

```
bhosts -s
RESOURCE                TOTAL       RESERVED       LOCATION
hspice                  76.0        0.0            host2
```

When LSF License Scheduler is configured to work with LSF Advanced Edition submission and execution clusters, LSF Advanced Edition considers LSF License Scheduler cluster mode and fast dispatch project mode features to be shared features. When you run the **bhosts -s** command from a host in the submission cluster, it shows no TOTAL and RESERVED tokens available for the local hosts in the submission cluster, but shows the number of available tokens for TOTAL and the number of used tokens for RESERVED in the execution clusters.

host_name ... | host_group ... | compute unit ...
Displays only information about the specified hosts. Do not use quotation marks to specify multiple hosts.

For host groups and compute units, the names of the member hosts are displayed instead of the name of the host group or compute unit. Do not use quotation marks to specify multiple host groups or compute units.

cluster_name
LSF multicluster capability only. Displays information about hosts in the specified cluster.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Output: Host-based default

Displays the following fields:

HOST_NAME
The name of the host. If a host has running batch jobs, but the host is removed from the configuration, the host name is displayed as `lost_and_found`.

For condensed host groups, the HOST_NAME value is the name of host group.

STATUS
With LSF multicluster capability, not shown for fully exported hosts.

The status of the host and the **sbatchd** daemon. Batch jobs can be dispatched only to hosts with an ok status. Host status has the following values:

ok
The host is available to accept batch jobs.

For condensed host groups, if a single host in the host group is ok, the overall status is also shown as ok.

If any host in the host group or compute unit is not ok, **bhosts** displays the first host status that it encounters as the overall status for the condensed host group. Use the **bhosts -X** command to see the status of individual hosts in the host group or compute unit.

unavail
The host is down, or LIM and the **sbatchd** daemon on the host are unreachable.

unreach
LIM on the host is running but the sbatchd daemon is unreachable.

closed
The host is not allowed to accept any remote batch jobs. The host can be closed for several reasons.

closed_Cu_excl
This host is a member of a compute unit that is running an exclusive compute unit job.

JL/U
With LSF multicluster capability, not shown for fully exported hosts.

The maximum number of job slots that the host can process on a per user basis. A dash (-) indicates no limit.

For condensed host groups or compute units, the JL/U value is the total number of job slots that all hosts in the group or unit can process on a per user basis.

The host does not allocate more than JL/U job slots for one user at the same time. These job slots are used by running jobs, as well as by suspended or pending jobs with reserved slots.

For preemptive scheduling, the accounting is different. These job slots are used by running jobs and by pending jobs with reserved slots.

MAX
The maximum number of job slots available. A dash (-) indicates no limit.

For condensed host groups and compute units, the MAX value is the total maximum number of job slots available in all hosts in the host group or compute unit.

These job slots are used by running jobs, as well as by suspended or pending jobs with reserved slots.

If preemptive scheduling is used, suspended jobs are not counted.

A host does not always have to allocate this many job slots if jobs are waiting. The host must also satisfy its configured load conditions to accept more jobs.

NJOBS
The number of tasks for all jobs that are dispatched to the host. The NJOBS value includes running, suspended, and chunk jobs.

For condensed host groups and compute units, the NJOBS value is the total number of tasks that are used by jobs that are dispatched to any host in the host group or compute unit.

If the `-alloc` option is used, total is the sum of the RUN, SSUSP, USUSP, and RSV counters.

RUN
The number of tasks for all running jobs on the host.

For condensed host groups and compute units, the RUN value is the total number of tasks for running jobs on any host in the host group or compute unit. If the `-alloc` option is used, total is the allocated slots for the jobs on the host.

SSUSP
The number of tasks for all system suspended jobs on the host.

For condensed host groups and compute units, the SSUSP value is the total number of tasks for all system suspended jobs on any host in the host group or compute unit. If the `-alloc` option used, total is the allocated slots for the jobs on the host.

USUSP
The number of tasks for all user suspended jobs on the host. Jobs can be suspended by the user or by the LSF administrator.

For condensed host groups and compute units, the USUSP value is the total number of tasks for all user suspended jobs on any host in the host group or compute unit. If the `-alloc` option used, total is the allocated slots for the jobs on the host.

RSV
The number of tasks for all pending jobs with reserved slots on the host.

For condensed host groups and compute units, the RSV value is the total number of tasks for all pending jobs with reserved slots on any host in the host group or compute unit. If the `-alloc` option used, total is the allocated slots for the jobs on the host.

Output: Host-based -l option

In addition to the default output fields, the `-l` option also displays the following information:

loadSched, loadStop
The scheduling and suspending thresholds for the host. If a threshold is not defined, the threshold from the queue definition applies. If both the host and the queue define a threshold for a load index, the most restrictive threshold is used.

The migration threshold is the time that a job dispatched to this host can remain suspended by the system before LSF attempts to migrate the job to another host.

STATUS

The long format that is shown by the `-l` option gives the possible reasons for a host to be closed. If a power policy is enabled in the `lsb.threshold` file, it shows the power state:

closed_Adm
The host is closed by the LSF administrator or root with the **badmin hclose** command. No job can be dispatched to the host, but jobs that are running on the host are not affected.

closed_Busy
The host is overloaded. At least one load index exceeds the configured threshold. Indices that exceed their threshold are identified by an asterisk (*). No job can be dispatched to the host, but jobs that are running on the host are not affected.

closed_Cu_Excl
This host is a member of a compute unit that is running an exclusive compute unit job (submitted with the **bsub -R "cu[excl]"** command).

closed_EGO
For EGO-enabled SLA scheduling, host is closed because it was not allocated by EGO to run LSF jobs. Hosts that are allocated from EGO display the status ok.

closed_Excl
The host is running an exclusive job (submitted with the **bsub -x** command).

closed_Full
The maximum number of job slots on the host was reached. No job can be dispatched to the host, but jobs that are running on the host are not affected.

closed_LIM
LIM on the host is unreachable, but the **sbatchd** daemon is running.

closed_Lock
The host is locked by the EGO administrator or root by using **lsadmin limlock** command. Running jobs on the host are suspended by EGO (SSUSP state). Use the **lsadmin limunlock** command to unlock LIM on the local host.

closed_Wind
The host is closed by a dispatch window that is defined in the `lsb.hosts` file. No job can be dispatched to the host, but jobs that are running on the host are not affected.

closed_RC
The LSF resource connector host is closed because of a resource provider reclaim request. Hosts are also marked as `closed_RC` before they are returned to a resource provider (such as EGO, OpenStack, Amazon Web Services) when maximum time-to-live (the **LSB_RC_EXTERNAL_HOST_MAX_TTL** parameter in the `lsf.conf` file) or host idle time (the **LSB_RC_EXTERNAL_HOST_IDLE_TIME** parameter in the `lsf.conf`file) was reached.

on
The host power state is on.

Note: Power state on does not mean that the host state is ok, which depends on whether the **lim** and **sbatchd** daemons can be connected by the master host.

off
The host is powered off by policy or manually.

suspend
The host is suspended by policy or manually with **badmin hpower**.

restarting
The host is resetting when resume operation failed.

resuming
The host is being resumed from standby state, which is triggered by either policy or cluster administrator.

suspending
The host is being suspended which is triggered by either policy or cluster administrator.

closed_Power
The host is put into power saving (suspend) state by the cluster administrator.

ok
Host suspend was triggered by power policy.

CPUF
Displays the CPU normalization factor of the host (see **lshosts(1)**).

DISPATCH_WINDOW
Displays the dispatch windows for each host. Dispatch windows are the time windows during the week when batch jobs can be run on each host. Jobs that are already started are not affected by the dispatch windows. When the dispatch windows close, jobs are not suspended. Jobs already running continue to run, but no new jobs are started until the windows reopen. The default for the dispatch window is no restriction or always open (that is, twenty-four hours a day and seven days a week). For the dispatch window specification, see the description for the DISPATCH_WINDOWS keyword under the -l option in the bqueues command.

CURRENT LOAD

Displays the total and reserved host load.

Reserved
You specify reserved resources by using the **bsub -R** option. These resources are reserved by jobs that are running on the host.

Total
The total load has different meanings, depending on whether the load index is increasing or decreasing.

For increasing load indices, such as run queue lengths, CPU usage, paging activity, logins, and disk I/O, the total load is the consumed plus the reserved amount. The total load is calculated as the sum of the current load and the reserved load. The current load is the load that is shown by the **lsload** command.

For decreasing load indices, such as available memory, idle time, available swap space, and available space in tmp, the total load is the available amount. The total load is the difference between the current load and the reserved load. This difference is the available resource as shown by the **lsload** command.

LOAD THRESHOLD

Displays the scheduling threshold (loadSched) and the suspending threshold (loadStop). Also displays the migration threshold if defined and the checkpoint support if the host supports checkpointing.

The format for the thresholds is the same as for batch job queues. For an explanation of the thresholds and load indices, see the description for the QUEUE SCHEDULING PARAMETERS keyword under the -l option of the **bqueues** command.

THRESHOLD AND LOAD USED FOR EXCEPTIONS

Displays the configured threshold of EXIT_RATE for the host and its current load value for host exceptions.

ADMIN ACTION COMMENT

If the EGO administrator specified an administrator comment with the -C option of the **badmin** host control commands **hclose** or **hopen**, the comment text is displayed.

PE NETWORK INFORMATION

Displays network resource information for IBM Parallel Edition (PE) jobs that are submitted with the bsub -network option, or to a queue (defined in the lsb.queuesfile) or an application profile (defined in the lsb.applications file) with the **NETWORK_REQ** parameter defined.

The following example shows PE NETWORK INFORMATION:

```
bhosts -l

...
PE NETWORK INFORMATION
NetworkID                       Status                  rsv_windows/total_windows
1111111                         ok                            4/64
2222222                         closed_Dedicated              4/64
...
```

NetworkID is the physical network ID returned by PE.

One of the following network Status values is displayed:

ok
Normal status.

closed_Full
All network windows are reserved.

closed_Dedicated
A dedicated PE job is running on the network (the usage=dedicated option is specified in the network resource requirement string).

unavail
Network information is not available.

CONFIGURED AFFINITY CPU LIST

The host is configured in the lsb.hosts file to accept jobs for CPU and memory affinity scheduling. If the **AFFINITY** parameter is configured as Y, the keyword all is displayed. If a CPU list is specified under the AFFINITY column, the configured CPU list for affinity scheduling is displayed.

Output: Resource-based -s option

The -s option displays the following resource information: the amounts that are used for scheduling, the amounts reserved, and the associated hosts for the resources. Only resources (shared or host-based) with numeric values are displayed.

The following fields are displayed:

RESOURCE
The name of the resource.

TOTAL
The total amount free of a resource that is used for scheduling.

RESERVED
The amount that is reserved by jobs. You specify the reserved resource by using the bsub -R option.

LOCATION
The hosts that are associated with the resource.

Output: Host-based -aff option

The -aff option displays host topology information for CPU and memory affinity scheduling. Only the topology nodes that contain CPUs in the list in the **CPULIST** parameter that is defined in the lsb.hosts file are displayed.

The following fields are displayed:

AFFINITY
If the host is configured in the lsb.hosts file to accept jobs for CPU and memory affinity scheduling, and the host supports affinity scheduling, AFFINITY: Enabled is displayed.

If the host is configured in the lsb.hosts file to accept jobs for CPU and memory affinity scheduling, but the host does not support affinity scheduling, AFFINITY: Disabled (not supported) is

displayed. If the host is LIM is not available or **sbatchd** is unreachable, `AFFINITY: UNKNOWN` is displayed.

Host[*memory*] *host_name*
Maximum available memory on the host. If memory availability cannot be determined, a dash (-) is displayed for the host. If the `-l` option is specified with the `-aff` option, the host name is not displayed.

For hosts that do not support affinity scheduling, a dash (-) is displayed for host memory and no host topology is displayed.

NUMA[*numa_node*: *requested_mem* / *max_mem*]
Requested and total NUMA node memory. It is possible for requested memory for the NUMA node to be greater than the maximum available memory displayed.

A socket is a collection of cores with a direct pipe to memory. Each socket contains 1 or more cores. A socket is not necessarily a physical socket, but rather refers to the memory architecture of the machine.

A core is a single entity capable of performing computations.

A node contains sockets. A socket contains cores, and a core can contain threads if the core is enabled for multithreading.

If no NUMA nodes are present, then the NUMA layer in the output is not shown. Other relevant items such as host, socket, core, and thread are still shown.

If the host is not available, only the host name is displayed. A dash (-) is shown where available host memory would normally be displayed.

The following example shows `CONFIGURED AFFINITY CPU LIST`:

```
bhosts -l -aff hostA
HOST  hostA
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV DISPATCH_WINDOW
ok              60.00     -      8      0      0      0      0      0      -

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem  slots
 Total           0.0   0.0   0.0   30%   0.0   193   25     0 8605M  5.8G 13.2G      8
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M      -


 LOAD THRESHOLD USED FOR SCHEDULING:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -


 CONFIGURED AFFINITY CPU LIST: all

 AFFINITY: Enabled
 Host[15.7G]
     NUMA[0: 100M / 15.7G]
         Socket0
             core0(0)
         Socket1
             core0(1)
         Socket2
             core0(2)
         Socket3
             core0(3)
         Socket4
             core0(4)
         Socket5
             core0(5)
         Socket6
             core0(6)
         Socket7
             core0(7)
```

When EGO detects missing elements in the topology, it attempts to correct the problem by adding the missing levels into the topology. In the following example, sockets and cores are missing on host `hostB`:

```
...
Host[1.4G] hostB
    NUMA[0: 1.4G / 1.4G] (*0 *1)
...
```

A job that requests two cores, or two sockets, or 2 CPUs runs. Requesting two cores from the same NUMA node runs. However, a job that requests two cores from the same socket remains pending.

Output: GPU-based -gpu option

The `-gpu` option displays information of the GPUs on the host.

The following fields are displayed:

HOST
The host name.

ID
The GPU IDs on the host. Each GPU is shown as a separate line.

MODEL
The full model name, which consists of the GPU brand name and the model type.

MUSED
The amount of GPU memory that is actually used by the job.

MRSV
The amount of GPU memory that is reserved by the job.

NJOBS
The total number of jobs that are using the GPUs.

RUN
The total number of running jobs that are using the GPUs.

SUSP
The total number of suspended jobs that are using the GPUs.

RSV
The total number of pending jobs that reserved the GPUs.

If the `-l` option is specified with the `-gpu` option, shows more details of the GPUs with the following fields:

NGPUS
The total number of GPUs on the host.

SHARED_AVAIL
The current total number of GPUs that are available for concurrent use by multiple jobs (that is, when the job is submitted with `-gpu mode=shared` or `-gpu j_exclusive=no` options)

EXCLUSIVE_AVAIL
The current total number of GPUs that are used exclusive by the job (that is, when the job is submitted with `-gpu mode=exclusive_process` or `-gpu j_exclusive=yes` options)

STATIC ATTRIBUTES
Static GPU information. The following field is specific to this section:

NVLINK
The connections with other GPUs on the same host.

The connection flag of each GPU is separated by a slash (/) with the next GPU, with a Y showing that there is a direct NVLink connection with that GPU.

DYNAMIC ATTRIBUTES
The latest GPU usage information as maintained by LSF.

GPU JOB INFORMATION
Information on jobs that are using the host's GPUs. The following fields are specific to this section:

JEXCL
Flag to indicate whether the GPU job requested that the allocated GPUs cannot by used by other jobs (that is, whether the job was submitted with `-gpu j_exclusive=yes`)

RUNJOBIDS
The IDs of the running GPU jobs on the GPU.

SUSPJOBIDS
The IDs of the suspended GPU jobs on the GPU.

RSVJOBIDS
The IDs of the pending GPU jobs that reserved the GPU.

Resource connector -rconly option

The `-rconly` option displays information that is specific to the LSF resource connector.

The following fields are displayed:

PUB_DNS_NAME and PUB_IP_ADDRESS
Public DNS name and IP address of the host.

PRIV_DNS_NAME and PRIV_IP_ADDRESS
Private DNS name and IP address of the host.

RC_STATUS
LSF resource connector status.

PROV_STATUS
Resource provider status.

TAG
The RC_ACCOUNT value that is defined in the `lsb.queues` or `lsb.applications` files.

UPDATED_AT
Time stamp of the latest status change.

INSTANCE_ID
ID of the created machine instance. This ID uniquely identifies the host in LSF.

For example,

```
bhosts -rconly
PROVIDER : aws
  TEMPLATE : aws-vm-1
    PUB_DNS_NAME       PUB_IP_ADDRESS  PRIV_DNS_NAME      PRIV_IP_ADDRESS
RC_STATUS             PROV_STATUS           TAG             UPDATED_AT              INSTANCE_ID
    ec2-52-43-171-109. 52.43.171.109   ip-192-168-0-85.us
192.168.0.85    Done                  terminated            default        2017-05-31T14:30:47CDT  -
    ec2-35-160-157-112 35.160.157.112  ip-192-168-0-69.us
192.168.0.69    Allocated             running               default        2017-05-31T14:32:00CDT  -
```

Files

Reads the `lsb.hosts` file.

See also

`lsb.hosts`, **bqueues**, **lshosts**, **badmin**, **lsadmin**

Chapter 16. bhpart

Displays information about host partitions

Synopsis

`bhpart` [`-r`] [*host_partition_name* ...]

`bhpart` [`-h` | `-V`]

Description

By default, displays information about all host partitions. Host partitions are used to configure host-partition fairshare scheduling.

Options

-r
Displays the entire information tree that is associated with the host partition recursively.

host_partition_name ...
Displays information about the specified host partitions only.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Output

The following fields are displayed for each host partition:

HOST_PARTITION_NAME
Name of the host partition.

HOSTS
Hosts or host groups that are members of the host partition. The name of a host group is appended by a slash (/). To see information about host groups, use the **bmgroup** command.

USER/GROUP
Name of users or user groups who have access to the host partition. To see information about user groups, use the **bugroup** command.

SHARES
Number of shares of resources that are assigned to each user or user group in this host partition, as configured in the `lsb.hosts` file. The shares affect dynamic user priority for when fairshare scheduling is configured at the host level.

PRIORITY
Dynamic user priority for the user or user group. Larger values represent higher priorities. Jobs belonging to the user or user group with the highest priority are considered first for dispatch.

In general, users or user groups with larger shares, fewer started and reserved jobs, and a lower CPU time and run time have higher priority.

STARTED
Number of job slots that are used by running or suspended jobs that are owned by users or user groups in the host partition.

RESERVED
Number of job slots that are reserved by the jobs that are owned by users or user groups in the host partition.

CPU_TIME
Cumulative CPU time that is used by jobs of users or user groups that are run in the host partition. Measured in seconds, to one decimal place.

LSF calculates the cumulative CPU time by using the actual (not normalized) CPU time and a decay factor. The decay factor is calculated such that 1 hour of recently used CPU time decays to 0.1 hours after an interval of time that is specified by the **HIST_HOURS** parameter in the `lsb.params` file (5 hours by default).

RUN_TIME
Wall-clock run time plus historical run time of jobs of users or user groups that are run in the host partition. Measured in seconds.

LSF calculates the historical run time by using the actual run time of finished jobs and a decay factor. The decay factor is calculated such that 1 hour of recently used run time decays to 0.1 hours after an interval of time that is specified by the **HIST_HOURS** parameter in the `lsb.params` file (5 hours by default). Wall-clock run time is the run time of running jobs.

ADJUST
Dynamic priority calculation adjustment that is made by the user-defined fairshare plug-in (`libfairshareadjust.*`).

The fairshare adjustment is enabled and weighted by the parameter **FAIRSHARE_ADJUSTMENT_FACTOR** in the `lsb.params` file.

Files

Reads `lsb.hosts`.

See also

bugroup, **bmgroup**, **lsb.hosts**

Chapter 17. bimages

Displays information on Docker container images

Synopsis

`bimages` [`-m` *host_name* ... | `-m` *host_group* ...] [`-o` "*field_name* ..." [`-json`]] [*image-reponame*[`:`*image-tagname*]]

`bimages` [`-m` *host_name* ... | `-m` *host_group* ...] [`-o` "*field_name* ..."] [`-noheader`] [*image-reponame*[`:`*image-tagname*]]

`bimages` [`-h` | `-V`]

Description

By default, returns the following information about all LSF-invoked Docker container images: container image name, size of the image file, how many pending job tasks will use the container image, LSF host name, how many running job tasks and container instances are using the container image.

Pending and running job tasks are not counted for container jobs with `UNKNOWN` status.

Job arrays are considered as multi-job elements in this command.

Options

-json
Displays the customized output in JSON format.

When specified, **bimages -o** displays the customized output in the JSON format.

This option only applies to output for the **bimages -o** command for customized output. This has no effect when running **bimages** without the `-o` option.

-m *host_name*... | -m *host_group* ...
Displays information on containers that are invoked by LSF on the specified hosts. If a host group is specified, **bimages** displays containers that are invoked by LSF on every host in the host group.

If `-m` is not specified, **bimages** displays containers that are invoked by LSF on every host in the cluster.

-noheader
Removes the column headings from the output.

When specified, **bimages** displays the values of the fields without displaying the names of the fields. This is useful for script parsing, when column headings are not necessary.

This option cannot be used with the `-json` option.

-o "*field_name* ..."
Sets the customized output format. Specify which **bimages** fields and in which order to display.

The following are the field names used to specify the **bimages** fields to display and units of measurement for the displayed field:

Table 2. Output fields for bimages

Field name	**Unit**
REPO	
TAGS	
IMAGE_NAME	

Table 2. Output fields for bimages (continued)	
Field name	**Unit**
SIZE	MB
DEMAND	
HOST_NAME	
NJOBS	
NCTNRS	
LAST_USED	time stamp
TOTAL_NJOBS	
PULLED_TIME	time stamp
IMAGE_ID	

Field names are space-separated and case-sensitive.

For example,

```
bimages -o "IMAGE_NAME SIZE HOST_NAME"
```

***image-reponame*[:*image-tagname*]**
Displays information about the specified container image. If there are no specified image tags, **bimages** *image-reponame* displays the latest container image.

If this argument is not specified, **bimages** displays all container images.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Default Output

Displays the following fields:

IMAGE_NAME
The name of the container image. This is a combination of the container image repository and tag names.

SIZE
The size of the container image file.

DEMAND
The number of pending job tasks that will use the container image. This number includes jobs in the `PENDING` and `PSUSPEND` states, but does not include jobs in the `FWD_PEND` or any other states.

HOST_NAME
Name of the LSF host.

NJOBS
The number of running job tasks that are using the container image. This number includes jobs in the `RUNNING`, `USUSPEND`, and `SSUSPEND` states.

NCTNRS
The number of running container instances that are using the container image on the host. This number includes container instances that are started by other applications.

Customized Output (-o)

In addition to the fields displayed for the default output, the `-o` option can display the following specified fields:

REPO
The name of the container image repository.

TAGS
The name of the container image tags.

LAST_USED
The time that the container image was last used by an LSF job. If a running job is using the container image, the value is the current time, otherwise, this is the time that the job ended. A hyphen (`-`) indicates that the container image was never used.

TOTAL_NJOBS
The total number of running job tasks that are using the container image on all hosts. This number is the sum of all NJOBS values on all LSF hosts with the same container image.

PULLED_TIME
The time that the container image was last pulled.

IMAGE_ID
The container image ID, in table format. This shows the first 12 bits of the container image ID, which is the same output format as the docker images command. When shown in JSON format, this displays the full image ID.

See also

LSF_IMAGE_INFO_PUBLISH_INTERVAL and **LSF_IMAGE_INFO_EXPIRE_INTERVAL** parameters in the `lsf.conf` file.

Chapter 18. bjdepinfo

Displays job dependencies.

Synopsis

`bjdepinfo` [-r *level*] [-l] [-p] *job_ID* | "*job_ID*[*index*]"

`bjdepinfo` -c [-r *level*] *job_ID* | "*job_ID*[*index*]"

`bjdepinfo` [-h] [-V]

Description

The command **bjdepinfo** displays all or selected job dependencies. You can get a list of other jobs that a job depends on (parent jobs) and jobs that depend on your job (child jobs). The command can also show if the job dependency condition was not satisfied.

Note:

The parent-child relationship does not indicate that one or more jobs are created by other jobs. A job dependency is when the start of a job depends on the status of other jobs.

Options

***job_ID* | "*job_ID*[*index*]"**

Required. Job ID of the job or job array on which to operate.

If you specify only a job ID for a job array, information about all jobs in the array is displayed.

Displays all jobs that this job depends on.

-r *level*

When combined with `-p`, prints the parent jobs that cause the current job to pend recursively.

When combined with `-c`, prints the child jobs that depend on the current job recursively.

When combined with `-l`, prints detailed parent job dependency information recursively. When combined with `-l` and `-p`, prints detailed information about the parent jobs that cause the current job to pend recursively.

In each case, you can specify the level by using a positive integer. Level indicates the number of degrees of separation from the original job.

For example, specify level 1 to see any jobs that directly depend on this job or that this job depends on. Specify level 2 if you also want to see all dependencies on the jobs that have a dependency on the originally specified job.

If the job was partially cleaned, an asterisk (*) is displayed before the status, and the job name is unavailable (-).

-l

For the job that you specify, shows the detailed parent job dependency information, including the condition of the job and whether a job's dependency requirements were satisfied.

-p

Shows the parent jobs that cause the current job to pend.

-c

Shows any child jobs of the job you specify and as any dependencies they have.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output

JOBID

The job ID of the job with a parent or child dependency.

PARENT

The job ID of the job that has other dependent jobs.

CHILD

The job ID of the job that depends on other jobs.

PARENT_STATUS

The status of the parent job listed. If the job was partially cleaned, an asterisk (*) is displayed before the status, and the job name is unavailable (-).

CHILD_STATUS

The status of the child job listed.

PARENT_NAME

The name of the parent job listed. If the job was partially cleaned, an asterisk (*) is displayed before the status, and the job name is unavailable (-).

CHILD_NAME

The name of the child job listed.

LEVEL

The degrees of separation of job dependencies. 1 means that a job is directly dependent; other numbers indicate the levels of indirect dependency.

Chapter 19. bjgroup

Displays information about job groups

Synopsis

`bjgroup [-N] [-s [group_name]]`

`bjgroup [-h | -V]`

Description

When LSF adds more resources to a running resizable job, the **bjgroups** command displays the added resources. When LSF removes resources from a running resizable job, the **bjgroups** command displays the updated resources.

Options

-s

Sorts job groups by group hierarchy.

For example, for job groups named /A, /A/B, /X and /X/Y, the **bjgroup** command without the -s option displays the following information:

```
bjgroup
GROUP_NAME         NJOBS   PEND    RUN    SSUSP  USUSP  FINISH  SLA   JLIMIT  OWNER
/A                 0       0       0      0      0      0       ()      0/10  user1
/X                 0       0       0      0      0      0       ()       0/-  user2
/A/B               0       0       0      0      0      0       ()       0/5  user1
/X/Y               0       0       0      0      0      0       ()       0/5  user2
```

For the same job groups, the **bjgroup -s** command displays the following information:

```
bjgroup -s
GROUP_NAME         NJOBS   PEND    RUN    SSUSP  USUSP  FINISH  SLA   JLIMIT  OWNER
/A                 0       0       0      0      0      0       ()       0/10  user1
/A/B               0       0       0      0      0      0       ()       0/5  user1
/X                 0       0       0      0      0      0       ()       0/-  user2
/X/Y               0       0       0      0      0      0       ()       0/5  user2
```

Specify a job group name to show the hierarchy of a single job group:

```
bjgroup -s /X
GROUP_NAME   NJOBS  PEND   RUN   SSUSP  USUSP  FINISH       SLA   JLIMIT  OWNER
/X             25     0    25       0      0       0   puccini  25/100   user1
/X/Y           20     0    20       0      0       0   puccini   20/30   user1
/X/Z            5     0     5       0      0       0   puccini    5/10   user2
```

Specify a job group name with a trailing slash character (/) to show only the root job group:

```
bjgroup -s /X/
GROUP_NAME   NJOBS  PEND   RUN   SSUSP  USUSP  FINISH      SLA   JLIMIT  OWNER
/X              25    0    25       0      0       0   puccini  25/100  user1
```

-N

Displays job group information by job slots instead of number of jobs. The values for NSLOTS, PEND, RUN, SSUSP, USUSP, RSV are all counted in slots rather than number of jobs:

```
bjgroup -N
GROUP_NAME NSLOTS PEND   RUN   SSUSP  USUSP   RSV      SLA     OWNER
/X             25    0    25       0      0     0  puccini     user1
/A/B           20    0    20       0      0     0   wagner     batch
```

The `-N` option by itself shows job slot info for all job groups. Combine with the `-s` option to sort the job groups by hierarchy:

```
bjgroup -N -s
GROUP_NAME NSLOTS PEND   RUN   SSUSP   USUSP  RSV      SLA     OWNER
/A              0    0     0       0       0    0   wagner      batch
/A/B            0    0     0       0       0    0   wagner      user1
/X             25    0    25       0       0    0   puccini     user1
/X/Y           20    0    20       0       0    0   puccini     batch
/X/Z            5     0    5       0       0    0   puccini     batch
```

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Default output

A list of job groups is displayed with the following fields:

GROUP_NAME
The name of the job group.

NJOBS
The current number of jobs in the job group. A parallel job is counted as 1 job, regardless of the number of job slots it uses.

PEND
The number of pending jobs in the job group.

RUN
The number of running jobs in the job group.

SSUSP
The number of system-suspended jobs in the job group.

USUSP
The number of user-suspended jobs in the job group.

FINISH
The number of jobs in the specified job group in EXITED or DONE state.

SLA
The name of the service class that the job group is attached to with **bgadd -sla** *service_class_name*. If the job group is not attached to any service class, empty parentheses `()` are displayed in the SLA name column.

JLIMIT
The job group limit set by **bgadd -L** or **bgmod -L**. Job groups that have no configured limits or no limit usage are indicated by a dash (-). Job group limits are displayed in a USED/LIMIT format. For example, if a limit of 5 jobs is configured and 1 job is started, **bjgroup** displays the job limit under JLIMIT as 1/5.

OWNER
The job group owner.

Example

```
bjgroup
```

```
GROUP_NAME  NJOBS  PEND   RUN   SSUSP  USUSP  FINISH      SLA   JLIMIT  OWNER
/fund1_grp      5     4     0       1      0      0   Venezia     1/5   user1
/fund2_grp     11     2     5       0      0      4   Venezia     5/5   user1
/bond_grp       2     2     0       0      0      0   Venezia     0/-   user2
/risk_grp       2     1     1       0      0      0        ()     1/-   user2
/admi_grp       4     4     0       0      0      0        ()     0/-   user2
```

Job slots (-N) output

The values for NSLOTS, PEND, RUN, SSUSP, USUSP, RSV are all counted in slots rather than number of jobs. A list of job groups is displayed with the following fields:

GROUP_NAME
The name of the job group.

NSLOTS
The total number of job slots held currently by jobs in the job group. This includes pending, running, suspended and reserved job slots. A parallel job that is running on *n* processors is counted as *n* job slots, since it takes *n* job slots in the job group.

PEND
The number of job slots used by pending jobs in the job group.

RUN
The number of job slots used by running jobs in the job group.

SSUSP
The number of job slots used by system-suspended jobs in the job group.

USUSP
The number of job slots used by user-suspended jobs in the job group.

RSV
The number of job slots in the job group that are reserved by LSF for pending jobs.

SLA
The name of the service class that the job group is attached to with the **bgadd -sla** *service_class_name* coammand. If the job group is not attached to any service class, empty parentheses () are displayed in the SLA name column.

OWNER

The job group owner.

Example

```
bjgroup -N
GROUP_NAME NSLOTS PEND   RUN   SSUSP  USUSP   RSV      SLA     OWNER
/X             25    0    25       0      0     0  puccini     user1
/A/B           20    0    20       0      0     0   wagner     batch
```

See also

bgadd, **bgdel**, **bgmod**

Chapter 20. bjobs

Displays and filters information about LSF jobs. Specify one or more job IDs (and, optionally, an array index list) to display information about specific jobs (and job arrays).

Synopsis

`bjobs` [*options*] [*job_ID* | "*job_ID*[*index_list*]" ...]

`bjobs -h[elp] [all] [description] [`*category_name* ...] [*-option_name* ...]

`bjobs -V`

Categories and options

Use the keyword `all` to display all options and the keyword `description` to display a detailed description of the **bjobs** command. For more details on specific categories and options, specify **bjobs -h** with the name of the categories and options.

Categories

Category: filter

Filter specific types of jobs: `-A`, `-aff`, `-app`, `-aps`, `-data`, `-fwd`, `-g`, `-G`, `-J`, `-Jd`, `-Lp`, `-m`, `-N`, `-P`, `-plan`, `-prio`, `-psum`, `-q`, `-rusage`, `-sla`, `-ss`, `-u`.

Category: format

Control the **bjobs** display format: `-aff`, `-cname`, `-hostfile`, `-l`, `-N`, `-noheader`, `-o`, `-psum`, `-sum`, `-UF`, `-w`, `-W`, `-WF`, `-WL`, `-WP`, `-X`.

Category: state

Display specific job states: `-a`, `-d`, `-N`, `-p`, `-pe`, `-pei`, `-pi`, `-psum`, `-r`, `-s`, `-sum`, `-x`.

Options

List of options for the **bjobs** command.

-A

Displays summarized information about job arrays.

Categories

filter

Synopsis

`bjobs -A`

Description

If you specify job arrays with the job array ID, and also specify `-A`, do not include the index list with the job array ID.

You can use `-w` to show the full array specification, if necessary.

-a

Displays information about jobs in all states, including jobs that finished recently.

Categories

state

Synopsis

```
bjobs -a
```

Description

The finished jobs that `-a` displays are those that finished within an interval specified by CLEAN_PERIOD in `lsb.params` (the default period is 1 hour).

Use `-a` with `-x` option to display all jobs that have triggered a job exception (overrun, underrun, idle).

Examples

```
bjobs -u all -a
```

Displays all jobs of all users.

-aff

Displays information about jobs with CPU and memory affinity resource requirements for each task in the job.

Categories

filter, format

Synopsis

```
bjobs -l | -UF [-aff]
```

Conflicting options

Use only with the `-l` or `-UF` option.

Description

If the job is pending, the requested affinity resources are displayed. For running jobs, the effective and combined affinity resource allocation decision made by LSF is also displayed, along with a table headed AFFINITY that shows detailed memory and CPU binding information for each task, one line for each allocated processor unit. For finished jobs (EXIT or DONE state), the affinity requirements for the job, and the effective and combined affinity resource requirement details are displayed.

Use **`bhist -l -aff`** to show the actual affinity resource allocation for finished jobs.

-app

Displays information about jobs submitted to the specified application profile.

Categories

filter

Synopsis

`bjobs -app` *application_profile_name*

Description

You must specify an existing application profile.

Examples

`bjobs -app fluent`

Displays all jobs belonging to the application profile `fluent`.

-aps

Displays absolute priority scheduling (APS) information for pending jobs in a queue with APS_PRIORITY enabled.

Categories

format

Synopsis

`bjobs -aps`

Description

The APS value is calculated based on the current scheduling cycle, so jobs are not guaranteed to be dispatched in this order.

Pending jobs are ordered by APS value. Jobs with system APS values are listed first, from highest to lowest APS value. Jobs with calculated APS values are listed next ordered from high to low value. Finally, jobs not in an APS queue are listed. Jobs with equal APS values are listed in order of submission time. APS values of jobs not in an APS queue are shown with a dash (-).

If queues are configured with the same priority, **`bjobs -aps`** may not show jobs in the correct expected dispatch order. Jobs may be dispatched in the order the queues are configured in `lsb.queues`. You should avoid configuring queues with the same priority.

For resizable jobs, `-aps` displays the latest APS information for running jobs with active resize allocation requests. LSF handles the dynamic priority for running jobs with active resize requests. The displayed job priority can change from time to time.

-cname

In IBM Spectrum LSF Advanced Edition, includes the cluster name for execution cluster hosts in the output.

Categories

format

Synopsis

`bjobs -cname`

Examples

```
% bjobs -l -cname
Job <1>, User <lsfuser>, Project <default>, Status <RUN>, Queue <queue1>,
Command <myjob>
Mon Nov 29 14:08:35: Submitted from host <hostA>, CWD </home/lsfuser>,
                     Re-runnable;
Mon Nov 29 14:08:38: Job <1> forwarded to cluster <cluster3>;
Mon Nov 29 14:08:44: Started on <hostC@cluster3>, Execution Home
                     </home/lsfuser>, Execution CWD </home/lsfuser>;
Mon Nov 29 14:08:46: Resource usage collected.
 MEM: 2 Mbytes;  SWAP: 32 Mbytes;  NTHREAD: 1
 PGID: 6395;  PIDs: 6395
```

```
SCHEDULING PARAMETERS:
           r15s r1m  r15m ut pg io ls it tmp swp mem
 loadSched -    -    -    -  -  -  -  -   -  -   -
 loadStop  -    -    -    -  -  -  -  -   -  -   -
 ...
```

-d

Displays information about jobs that finished recently.

Categories

state

Synopsis

`bjobs -d`

Description

The finished jobs that `-d` displays are those that finished within an interval specified by CLEAN_PERIOD in `lsb.params` (the default period is 1 hour).

Examples

```
bjobs -d -q short -m hostA -u user1
```

Displays all the recently finished jobs submitted by `user1` to the queue `short`, and executed on the host `hostA`.

-data

Displays the data file requirements for the job. The `-data` option acts as a filter to show only jobs with data requirements. For example, the option lists jobs that are submitted with `-data`. The option also lists files or data requirement tags that are requested by the job.

Categories

filter

Synopsis

`bjobs -data` [*jobID*]

Conflicting options

Do not use the `-data` option with the following options: `-A`, `-sum`.

Description

The following units are displayed for file size:

- *nnn* B if file size is less than 1 KB
- *nnn*`[.`*n*`]` KB if file size is less than 1 MB
- *nnn*`[.`*n*`]` MB if file size is less than 1 GB
- *nnn*`[.`*n*`]` GB if file size is 1 GB or larger
- *nnn*`[.`*n*`]` EB if file size is 1 EB or larger

A dash (`-`) indicates that a size or modification time stamp value is not available.

For jobs with data requirements that are specified as tags, the `-data` option shows the tag names.

Examples

```
bjobs -data 1962
JOBID   USER    STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME   SUBMIT_TIME
1962    user1   PEND  normal hostA                *p 1000000 Sep 20 16:31
FILE                              SIZE   MODIFIED
datahost:/proj/user1/input1.dat      500 M   Jun 27 16:37:52
datahost:/proj/user1/input2.dat      100 M   Jun 27 16:37:52
datahost:/proj/user1/input3.dat      -      -
```

For jobs with data requirements specified as tags, the `-data` option shows the tag names:

```
bjobs -data 1962
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1962    user1   PEND  normal     hostA                  *p 1000000 Sep 20 16:31
TAG
OUTPUT_FROM_J1
OUTPUT_FROM_J2
```

For jobs with a folder as data requirement, the `-data` shows the folder name. Individual files in the folder are not shown. For example, for the following data requirement

```
bsub -data "/home/user/folder1/" myjob
```

the `-data` option shows the folder name `/home/user/folder1/`:

```
bjobs -data 44843
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
44843   user1   PEND  normal     hosta                  myjob      May 10 14:42

 FILE                                              SIZE         MODIFIED
 hosta:/home/user1/folder1/                        4 KB         May  9 09:53
```

-fwd

In LSF multicluster capability job forwarding mode, filters output to display information on forwarded jobs.

Categories

filter

Synopsis

`bjobs -fwd`

Conflicting options

Do not use with the following options: `-A`, `-d`, `-sla`, `-ss`, `-x`.

Description

In LSF multicluster capability job forwarding mode, filters output to display information on forwarded jobs, including the forwarded time and the name of the cluster to which the job was forwarded. **-fwd** can be used with other options to further filter the results. For example, **bjobs -fwd -r** displays only forwarded running jobs.

To use **-x** to see exceptions on the execution cluster, use **bjobs -m** *execution_cluster* **-x**.

Examples

```
% bjobs -fwd
JOBID USER    STAT QUEUE  EXEC_HOST JOB_NAME   CLUSTER  FORWARD_TIME
123   lsfuser RUN  queue1 hostC     sleep 1234 cluster3 Nov 29 14:08
```

-G

Displays jobs associated with the specified user group.

Categories

filter

Synopsis

`bjobs -G` *user_group*

Conflicting options

Do not use with the `-u` option.

Description

Only displays jobs associated with a user group submitted with **bsub -G** for the specified user group. The –G option does not display jobs from subgroups within the specified user group. Jobs associated with the user group at submission are displayed, even if they are later switched to a different user group.

You can only specify a user group name. The keyword `all` is not supported for `-G`.

-g

Displays information about jobs attached to the specified job group.

Categories

filter

Synopsis

`bjobs -g` *job_group_name*

Description

Use `-g` with `-sla` to display job groups attached to a time-based service class. Once a job group is attached to a time-based service class, all jobs submitted to that group are subject to the SLA.

bjobs -l with **-g** displays the full path to the group to which a job is attached.

Examples

```
bjobs -g /risk_group
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
113     user1   PEND  normal     hostA                   myjob     Jun 17 16:15
111     user2   RUN   normal     hostA       hostA       myjob     Jun 14 15:13
110     user1   RUN   normal     hostB       hostA       myjob     Jun 12 05:03
104     user3   RUN   normal     hostA       hostC       myjob     Jun 11 13:18
```

To display the full path to the group to which a job is attached, run **bjobs -l -g**:

```
bjobs -l -g /risk_group
Job <101>, User <user1>, Project <default>, Job Group </risk_group>,
    Status <RUN>, Queue <normal>, Command <myjob>
Tue Jun 17 16:21:49 2009: Submitted from host <hostA>, CWD </home/user1;
Tue Jun 17 16:22:01 2009: Started on <hostA>;
...
```

-gpu

bjobs -l -gpu shows the following information on GPU job allocation:

Categories

filter

Synopsis

`bjobs -l | -UF [-gpu]`

Conflicting options

Use only with the `-l` or `-UF` option.

Description

Host Name
The name of the host.

GPU IDs on the host
Each GPU is shown as a separate line.

TASK and ID
List of job tasks and IDs using the GPU (separated by comma if used by multiple tasks)

MODEL
Contains the GPU brand name and model type name.

MTOTAL
The total GPU memory size.

GPU Compute Capability

MRSV
GPU memory reserved by the job

SOCKET
socket ID of the GPU located at

NVLINK
Indicates if the GPU has NVLink connections with other GPUs allocated for the job (ranked by GPU ID and including itself). The connection flag of each GPU is a character separated by "/" with the next GPU:

A "Y" indicates there is a direct NVLINK connection between two GPUs.

An "N" shows there is no direct NVLINK connection with that GPU.

A "-" shows the GPU is itself.

If the job exited abnormally due to a GPU-related error or warning, the error or warning message displays. If LSF could not get GPU usage information from DCGM, a hyphen (-) displays.

-hms

Displays times in the customized output in *hh:mm:ss* format.

Categories

format

Synopsis

`bjobs -o format [-json] -hms`

Conflicting options

Use only with the -o and -o -json options.

Description

When specified, **bjobs** displays any times (but not time stamps) in the customized output in *hh:mm:ss* format.

The cpu_used field normally uses one decimal place in the **bjobs** output, but if you specify -hms, **bjobs** rounds up this field to the next second. For example, if the cpu_used field is 0.2 seconds, specifying -hms rounds this number up to 00:00:01.

For the runtimelimit field, if the **ABS_RUNLIMIT** parameter is defined as Y in the lsb.params file and you specify -hms, **bjobs** does not display the host name. That is, if ABS_RUNLIMIT=Y is defined in the lsb.params file, **bjobs** normally displays the host name after the runtime limit (for example, 3.0/hostA). If you specify -hms, **bjobs** displays the runtime limit without the host name (for example, 00:03:00). This also applies if the **ABS_RUNLIMIT** parameter is defined as Y in the lsb.applications file and you specify -hms to a job that is submitted to an application profile with ABS_RUNLIMIT=Y.

Specifying the -hms option overrides the **LSB_HMS_TIME_FORMAT** environment variable, which overrides the **LSB_HMS_TIME_FORMAT** parameter setting in the lsf.conf file.

This option applies only to output for the **bjobs -o** and **bjobs -o -json** commands for customized output. This option has no effect when run with **bjobs** without the -o option.

-hostfile

Displays information about a job submitted with a user-specified host file.

Categories

format

Synopsis

bjobs -l | -UF [-hostfile]

Conflicting options

Use only with the -l or -UF option.

Description

If a job was submitted with **bsub -hostfile** or modified with **bmod -hostfile** to point to a user-specified host file, use -hostfile to show the user-specified host file path as well as the contents of the host file.

Use -hostfile together with -l or -UF, to view the user specified host file content as well as the host allocation for a given job.

Example

Use -l -hostfile to display a user-specified host file that was submitted with a job or added to a job.

For example:

```
bjobs -l -hostfile  2012
Job <2012>, User <userG>, Project <myproject>, Status <PEND>, Queue
                          <normal>, Commnad <sleep 10000>
                          Thu Aug 1 12:43:25: Submitted from host <host10a>,
                          CWD <$HOME>,Host file </home/userG/myhostfile>;

......
```

```
USER-SPECIFIED HOST FILE:
HOST            SLOTS
host01            3
host02            1
host01            1
host02            2
host03            1
```

-Jd

Displays information about jobs with the specified job description.

Categories

filter

Synopsis

`bjobs -Jd` *job_description*

Description

Only displays jobs that were submitted by the user running this command.

The job description can be up to 4094 characters long. Job descriptions are not unique.

The wildcard character (*) can be used anywhere within a job description.

-json

Displays the customized output in JSON format.

Categories

format

Synopsis

`bjobs -o` *format* `-json`

Conflicting options

Do not use with the `-noheader` option.

Description

When specified, **bjobs -o** displays the customized output in the JSON format.

This option applies only to output for the **bjobs -o** command for customized output. This option has no effect when run with **bjobs** without the `-o` option and the **LSB_BJOBS_FORMAT** environment variable or parameter are not defined.

-Lp

Displays jobs that belong to the specified LSF License Scheduler project.

Categories

filter

Synopsis

`bjobs -Lp` *ls_project_name*

-l

Long format. Displays detailed information for each job in a multi-line format.

Categories

format

Synopsis

```
bjobs -l
```

Description

The `-l` option displays the following additional information:

- Project name
- Job command
- Current working directory on the submission host
- Initial checkpoint period
- Checkpoint directory
- Migration threshold
- Predicted job start time
- Pending and suspending reasons
- Job status
- Job kill reason
- Resource usage
- Resource usage limits information
- Runtime resource usage information on the execution hosts
- Pending time limits, eligible pending time limits
- Job description
- Name of any esub/epsub used with the job
- Energy usage (if energy accounting with IBM Spectrum LSF Explorer is enabled by setting `LSF_QUERY_ES_FUNCTIONS="energy"` or `"all"` in the `lsf.conf` file)
- Approximate accumulated job disk usage (I/O) data on IBM Spectrum Scale (if IBM Spectrum Scale I/O accounting with IBM Spectrum LSF Explorer is enabled by setting `LSF_QUERY_ES_FUNCTIONS="gpfsio"` or `"all"` in the `lsf.conf` file)
- Planned start time for jobs with a schedule and reservation plan.

If the job was submitted with the **bsub -K** command, the `-l` option displays `Synchronous Execution`.

Use the **bjobs -A -l** command to display detailed information for job arrays, including job array job limit (`%` *job_limit*) if set.

Use the **bjobs -ss -l** command to display detailed information for session scheduler jobs.

Use the **bjobs -data -l** command to display detailed information for jobs with data requirements (for example, jobs that are submitted with `-data`).

The `bjobs -pl` command displays detailed information about all pending jobs of the invoker.

If the **JOB_IDLE** parameter is configured in the queue, use **bjobs -l** to display job idle exception information.

If you submitted your job with the `-U` option to use advance reservations that are created with the **brsvadd** command, **bjobs -l** shows the reservation ID used by the job.

If the **LSF_HPC_EXTENSIONS="SHORT_PIDLIST"** parameter is specified in the `lsf.conf` file, the output from **bjobs** is shortened to display only the first PID and a count of the process group IDs (PGIDs) and process IDs for the job. Without **SHORT_PIDLIST**, all of the process IDs (PIDs) for a job are displayed.

If the **LSF_HPC_EXTENSIONS="HOST_RUSAGE"** parameter is specified in the `lsf.conf` file, the output from the **bjobs -l** command reports the correct rusage-based usage and the total rusage that is being charged to the execution host.

If you submitted a job with multiple resource requirement strings by using the **bsub** `-R` option for the order, same, rusage, and select sections, **bjobs -l** displays a single, merged resource requirement string for those sections, as if they were submitted by using a single `-R` option.

If you submitted a job by using the OR (||) expression to specify alternative resources, this option displays the Execution rusage string with which the job runs.

Predicted start time for PEND reserve job is not shown with the **bjobs -l** command option. LSF does not calculate the predicted start time for PEND reserve job if no back fill queue is configured in the system. In that case, resource reservation for PEND jobs works as normal, and no predicted start time is calculated.

For resizable jobs, the `-l` option displays active pending resize allocation requests, and the latest job priority for running jobs with active pending resize requests.

For jobs with user-based fairshare scheduling, displays the charging SAAP (share attribute account path).

For jobs submitted to an absolute priority scheduling (APS) queue, `-l` shows the `ADMIN` factor value and the system APS value if they are not set by the administrator for the job.

For jobs submitted with SSH X11 forwarding, displays that the job was submitted in SSH X11 forwarding mode as well as the SSH command submitted (set in **LSB_SSH_XFORWARD_CMD** in `lsf.conf`).

If the job was auto-attached to a guarantee SLA, `-l` displays the auto-attached SLA name.

Specified CWD shows the value of the **bsub -cwd** option or the value of **LSB_JOB_CWD**. The CWD path with pattern values is displayed. CWD is the submission directory where **bsub** ran. If specified CWD was not defined, this field is not shown. The execution CWD with pattern values is always shown.

If the job was submitted with an energy policy, to automatically select a CPU frequency, `-l` shows the Combined CPU frequency (the CPU frequency that is selected for the job based on the energy policy tag, energy policy, and threshold file). If the job was submitted with a user-defined CPU frequency (by using **bsub -freq**), `-l` shows the Specified CPU frequency for the job.

For jobs submitted with the default GPU requirements (with the option `-gpu -`), use the **bjobs -l** command to see the default job-level resource requirement without details like `<num=1...>: Requested GPU`.

If the `-gpu` option specifies GPU requirements (for example, `-gpu num=3`, the **bjobs -l** shows the details as `Requested GPU <num=3>`.

The **bjobs -l** command displays an output section for GPU jobs that shows the combined and effective GPU requirements that are specified for the job. The GPU requirement string is in the format of the **GPU_REQ** parameter in the application profile or queue:

- The combined GPU requirement is merged based on the current GPU requirements in job, queue, application, or default GPU requirement.
- The effective GPU requirement is the one used by a started job. It never changes after a job is started.

For example,

```
bjobs -l
Job <101>, User <user1>, Project <default>, Status <RUN>, Queue <normal>, Co
                     mmand <sleep 10000>, Share group charged </user1>
Wed Jul 12 04:51:00: Submitted from host <hosta>, CWD </home/user1/,
                     Specified Hosts <hosta>, Requested GPU;
...
...

EXTERNAL MESSAGES:
```

```
MSG_ID FROM       POST_TIME      MESSAGE                                ATTACHMENT
0      user1      Jul 12 04:51   hosta:gpus=2,0,1;                           N

RESOURCE REQUIREMENT DETAILS:
Combined: select[type == any] order[r15s:pg] rusage[ngpus_physical=2.00]
Effective: select[type == any] order[r15s:pg] rusage[ngpus_physical=3.00]

GPU REQUIREMENT DETAILS:
Combined: num=2:mode=shared:mps=no:j_exclusive=no
Effective: num=3:mode=shared:mps=no:j_exclusive=no
```

Jobs that are submitted with an **esub** (or **epsub**) by using **bsub -a** (or modified by using **bmod -a**), shows the latest **esubs** used for execution in **bjobs -l** output, first with the default and then user **esubs**. If a user-specified **esub** script is the same as the default **esub** script, the duplicate **esubs** shows as one entry. If a job is submitted with an **esub** containing parameters, the **esub** and its parameters are shown in **bjobs -l** as well, and the format of the **esub** is the same as the **esub** that is specified in the job submission. For example:

```
bsub -a "test(a,b,c) " sleep 10000
```

is shown as:

```
Job <1561>, User <joes>, Project <default>, Status <RUN>, Queue <normal>,
Command <sleep 1000000>, Share group charged </joes>, Esub <test(a,b,c)>
```

If enhanced energy accounting with IBM Spectrum LSF Explorer is enabled (with **LSF_ENABLE_BEAT_SERVICE** in `lsf.conf`), output shows the energy usage in Joule and kWh.

If the `lsb.params` configuration file is configured with `ALLOCATION_PLANNER = Y`, and sets of candidate jobs have been identified for consideration for an allocation plan, LSF creates a scheduling and reservation allocation plan.

- **bjobs -l** : displays the planned start time for all jobs with an allocation plan.
- **bjobs -l -plan** : filter for jobs with allocation plans, displaying the planned start time and planned allocation for each job.

-m

Displays jobs dispatched to the specified hosts.

Categories

filter

Synopsis

`bjobs -m` *host_name* ... | `-m` *host_group* ... | `-m` *cluster_name* ...

Description

To see the available hosts, use **bhosts**.

If a host group or compute unit is specified, displays jobs dispatched to all hosts in the group. To determine the available host groups, use **bmgroup**. To determine the available compute units, use **bmgroup -cu**.

With LSF multicluster capability, displays jobs in the specified cluster. If a remote cluster name is specified, you see the remote job ID, even if the execution host belongs to the local cluster. To determine the available clusters, use **bclusters**.

Examples

```
bjobs -d -q short -m hostA -u user1
```

Displays all the recently finished jobs submitted by `user1` to the queue `short`, and executed on the host `hostA`.

-N

Displays information about done and exited jobs, also displays the normalized CPU time consumed by the job.

Categories

filter, format, state

Synopsis

`bjobs -N` *host_name* | `-N` *host_model* | `-N` *cpu_factor*

Description

Normalizes using the CPU factor specified, or the CPU factor of the host or host model specified.

Use with `-p`, `-r`, and `-s` to show information about pending, running, and suspended jobs along with done and exited jobs.

-noheader

Removes the column headings from the output.

Categories

format

Synopsis

```
bjobs -noheader
```

Description

When specified, **bjobs** displays the values of the fields without displaying the names of the fields. This is useful for script parsing, when column headings are not necessary.

This option applies to output for the **bjobs** command with no options, and to output for all **bjobs** options with short form output except for `-aff`, `-l`, `-UF`, `-N`, **-h**, and **-V**.

-o

Sets the customized output format.

Categories

format

Synopsis

`bjobs -o "`*field_name*`[:[-][`*output_width*`]] ... [delimiter='`*character*`']"`

`bjobs -o '`*field_name*`[:[-][`*output_width*`]] ... [delimiter="`*character*`"]'`

Description

- Specify which **bjobs** fields (or aliases instead of the full field names), in which order, and with what width to display.
- Specify only the **bjobs** field name or alias to set its output to unlimited width and left justification.
- Specify the colon (`:`) without a width to set the output width to the recommended width for that field.
- Specify the colon (`:`) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **bjobs** truncates the output:

  - For the JOB_NAME field, **bjobs** removes the header characters and replaces them with an asterisk (*)
  - For other fields, **bjobs** truncates the ending characters
- Specify a hyphen (-) to set right justification when **bjobs** displays the output for the specific field. If not specified, the default is to set left justification when **bjobs** displays the output for a field.
- Use `delimiter=` to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

To specify special delimiter characters in a csh environment (for example, $), use double quotation marks (") in the delimiter specification and single quotation marks (') in the `-o` statement:

`bjobs ... -o '`*`field_name`*`[:[-][`*`output_width`*`]] ... [delimiter="`*`character`*`"]'`

The `-o` option applies only to output for certain **bjobs** options:

- This option applies to output for the **bjobs** command with no options, and for **bjobs** options with short form output that filter information, including the following options: `-a`, `-app`, `-cname`, `-d`, `-g`, `-G`, `-J`, `-Jd`, `-Lp`, `-m`, `-P`, `-q`, `-r`, `-sla`, `-u`, `-x`, `-X`.
- This option applies to output for **bjobs** options that use a modified format and filter information, including the following options: `-fwd`, `-N`, `-p`, `-s`.
- This option does not apply to output for **bjobs** options that use a modified format, including the following options: `-A`, `-aff`, `-aps`, `-l`, `-UF`, `-ss`, `-sum`, `-UF`, `-w`, `-W`, `-WF`, `-WL`, `-WP`.

The **bjobs -o** option overrides the **LSB_BJOBS_FORMAT** environment variable, which overrides the **LSB_BJOBS_FORMAT** setting in `lsf.conf`.

The following are the field names used to specify the **bjobs** fields to display, recommended width, aliases you can use instead of field names, and units of measurement for the displayed field:

Table 3. Output fields for bjobs

Field name	Width	Aliases	Unit	Category
jobid	7	id		Common
jobindex	8			
stat	5			
user	7			
user_group	15	ugroup		
queue	10			
job_name	10	name		
job_description	17	description		
proj_name	11	proj, project		
application	13	app		
service_class	13	sla		
job_group	10	group		
job_priority	12	priority		
rsvid	40			
esub	20			
kill_reason	50			
dependency	15			
pend_reason	11			
charged_saap	50	saap		
command	15	cmd		Command
pre_exec_command	16	pre_cmd		
post_exec_command	17	post_cmd		
resize_notification_command	27	resize_cmd		
pids	20			
exit_code	10			
exit_reason	50			
interactive	11			

Table 3. Output fields for bjobs (continued)

Field name	Width	Aliases	Unit	Category
from_host	11			Host
first_host	11			
exec_host	11			
nexec_host **Note:** If the allocated host group or compute unit is condensed, this field does not display the real number of hosts. Use **bjobs -X -o** to view the real number of hosts in these situations.	10			
ask_hosts	30			
alloc_slot	20			
nalloc_slot	10			
host_file	10			
exclusive	13			
nreq_slot	10			

Table 3. Output fields for bjobs (continued)

Field name	Width	Aliases	Unit	Category
submit_time	15		time stamp	Time
start_time	15		time stamp	
estimated_start_time	20	estart_time	time stamp	
specified_start_time	20	sstart_time	time stamp	
specified_terminate_time	24	sterminate_time	time stamp	
time_left	11		seconds	
finish_time	16		time stamp	
estimated_run_time	20	ertime	seconds	
ru_utime	12		seconds	
ru_stime	12		seconds	
%complete	11			
warning_action	15	warn_act		
action_warning_time	19	warn_time		
pendstate (IPEND/EPEND/ NOTPEND)	9			
pend_time	12		seconds	
ependtime	12		seconds	
ipendtime	12		seconds	
estimated_sim_start_time	24	esstart_time	time stamp	
effective_plimit (run with **bjobs -p** to show information for pending jobs only)	18		seconds	
plimit_remain (run with **bjobs -p** to show information for pending jobs only) A negative number indicates the amount of time in which the job exceeded the pending time limit, while a positive number shows that the time remaining until the pending time limit is reached.	15		seconds	
effective_eplimit (run with **bjobs -p** to show information for pending jobs only)	19		seconds	
eplimit_remain (run with **bjobs -p** to show information for pending jobs only)	16		seconds	

Table 3. Output fields for bjobs (continued)

Field name	Width	Aliases	Unit	Category
cpu_used	10			CPU
run_time	15		seconds	
idle_factor	11			
exception_status	16	except_stat		
slots	5			
mem	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)	
max_mem	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)	
avg_mem	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)	
memlimit	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)	
swap	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)	
swaplimit	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)	
gpu_num	10	gnum		GPU
gpu_mode	20	gmode		
j_exclusive	15	j_excl		
gpu_alloc	30	galloc		
nthreads	10			Resource usage
hrusage	50			
min_req_proc	12			Resource requirement
max_req_proc	12			
effective_resreq	17	eresreq		
combined_resreq	20	cresreq		
network_req	15			

Table 3. Output fields for bjobs (continued)

Field name	Width	Aliases	Unit	Category
filelimit	10			Resource limits
corelimit	10			
stacklimit	10			
processlimit	12			
runtimelimit	12			
plimit	10		seconds	
eplimit	10		seconds	
input_file	10			File
output_file	11			
error_file	10			
output_dir	15			Directory
sub_cwd	10			
exec_home	10			
exec_cwd	10			
licproject	20			License
forward_cluster	15	fwd_cluster		MultiCluster
forward_time	15	fwd_time	time stamp	
srcjobid	8			
dstjobid	8			
source_cluster	15	srcluster		
energy			Joule	Energy
gpfsio Job disk usage (I/O) data on IBM Spectrum Scale.				

Field names and aliases are not case-sensitive. Valid values for the output width are any positive integer 1 - 4096. If the jobid field is defined with no output width and **LSB_JOBID_DISP_LENGTH** is defined in `lsf.conf`, the **LSB_JOBID_DISP_LENGTH** value is used for the output width. If jobid is defined with a specified output width, the specified output width overrides the **LSB_JOBID_DISP_LENGTH** value.

Example

```
bjobs -o "jobid stat: queue:- project:10 application:-6 delimiter='^'" 123
```

This command (used to illustrate the different subcommands for `-o`) displays the following fields for a job with the job ID 123:

- JOBID with unlimited width and left-aligned. If LSB_JOBID_DISP_LENGTH is specified, that value is used for the output width instead.
- STAT with a maximum width of 5 characters (which is the recommended width) and left-aligned.
- QUEUE with a maximum width of 10 characters (which is the recommended width) and right-aligned.

- PROJECT with a maximum width of 10 characters and left-aligned.
- APPLICATION with a maximum width of 6 characters and right-aligned.
- The ^ character is displayed between different headers and fields.

-P

Displays jobs that belong to the specified project.

Categories

filter

Synopsis

`bjobs -P` *project_name*

-p

Displays pending jobs, together with the pending reasons that caused each job not to be dispatched during the last dispatch turn.

Categories

state

Synopsis

`bjobs -p`*<level0-3>*

Description

Displays the pending reason or reasons and the number of hosts giving that reason.

- 0: Displays **bjobs -p** output as before the LSF 10.1 release.
- 1: Displays the single key pending reason.
- 2: Displays categorized host-based pending reasons for candidate hosts in the cluster. For the candidate hosts, the actual reason on each host is shown. For each pending reason, the number of hosts that give the reason is shown. The actual pending reason messages appear from most to least common.
- 3: Displays categorized host-based pending reasons for both candidate and non-candidate hosts in the cluster. For both the candidate and non-candidate hosts, the actual pending reason on each host is shown. For each pending reason, the number of hosts that show that reason is given. The actual reason messages appear from most to least common.

If no level is specified, the default is shown, as set by the `LSB_BJOBS_PENDREASON_LEVEL` parameter in the `lsf.conf` file.

With IBM Spectrum LSF License Scheduler (LSF License Scheduler), the pending reason also includes the project name for project mode or fast dispatch mode, and the cluster name for cluster mode features. Jobs with an invalid project name show the project name as a hyphen (`-`). If a default project is configured for that feature, it shows `default` as the project name.

If `-l` is also specified, the pending reason shows the names of the hosts and any hosts that are on the job's host exclusion list.

With the LSF multicluster capability, **-l** shows the names of hosts in the local cluster.

Each pending reason is associated with one or more hosts and it states the cause why these hosts are not allocated to run the job. In situations where the job requests specific hosts (using **bsub -m**), users may see reasons for unrelated hosts also being displayed, together with the reasons associated with the requested hosts.

In the case of host-based pre-execution failure, pending reasons will be displayed.

The life cycle of a pending reason ends after the time indicated by **PEND_REASON_UPDATE_INTERVAL** in `lsb.params`.

When the job slot limit is reached for a job array (**bsub -J "jobArray[indexList] %job_slot_limit"**) the following message is displayed:

```
The job array has reached its job slot limit.
```

Examples

```
bjobs -pl
```

Displays detailed information about all pending jobs of the invoker. Also displays the names of any hosts that are in the job's host exclusion list (that is, hosts that are excluded from the job).

```
bjobs -ps
```

Display only pending and suspended jobs.

-pe

Displays pending jobs that are eligible for scheduling.

Categories

state

Synopsis

```
bjobs -pe
```

Description

A job that is in an eligible pending state is a job that LSF would normally select for resource allocation, but is currently pending because its priority is lower than other jobs. It is a job that is eligible for scheduling and will be run if there are sufficient resources to run it. In addition, for chunk jobs in `WAIT` status, the time spent in the `WAIT` status is counted as eligible pending time.

This option is valid only when **TRACK_ELIGIBLE_PENDINFO** in `lsb.params` is set to Y or y (enabled).

-pei

Displays pending jobs divided into lists of jobs that are eligible for scheduling and ineligible for scheduling.

Categories

state

Synopsis

```
bjobs -pei
```

Description

A job that is in an eligible pending state is a job that LSF would normally select for resource allocation, but is currently pending because its priority is lower than other jobs. It is a job that is eligible for scheduling and will be run if there are sufficient resources to run it.

An ineligible pending job remains pending even if there are enough resources to run it and is therefore ineligible for scheduling. Reasons for a job to remain pending, and therefore be in an ineligible pending state, include the following:

- The job has a start time constraint (specified with the `-b` option)
- The job is suspended while pending (in a PSUSP state).

- The queue of the job is made inactive by the administrator or by its time window.
- The job's dependency conditions are not satisfied.
- The job cannot fit into the run time window (**RUN_WINDOW**)
- Delayed scheduling is enabled for the job (**NEW_JOB_SCHED_DELAY** is greater than zero)
- The job's queue or application profile does not exist.

A job that is not under any of the ineligible pending state conditions is treated as an eligible pending job. In addition, for chunk jobs in `WAIT` status, the time spent in the `WAIT` status is counted as eligible pending time.

This option is valid only when **TRACK_ELIGIBLE_PENDINFO** in `lsb.params` is set to Y or y (enabled).

-pi

Displays pending jobs that are ineligible for scheduling.

Categories

state

Synopsis

```
bjobs -pi
```

Description

An ineligible pending job remains pending even if there are enough resources to run it and is therefore ineligible for scheduling. Reasons for a job to remain pending, and therefore be in an ineligible pending state, include the following:

- The job has a start time constraint (specified with the `-b` option)
- The job is suspended while pending (in a PSUSP state).
- The queue of the job is made inactive by the administrator or by its time window.
- The job's dependency conditions are not satisfied.
- The job cannot fit into the run time window (**RUN_WINDOW**)
- Delayed scheduling is enabled for the job (**NEW_JOB_SCHED_DELAY** is greater than zero)
- The job's queue or application profile does not exist.

A job that is not under any of the ineligible pending state conditions is treated as an eligible pending job.

This option is valid only when **TRACK_ELIGIBLE_PENDINFO** in `lsb.params` is set to Y or y (enabled).

-plan

Filter for the PEND jobs that have an allocation plan. See **ALLOCATION_PLANNER** (lsb.params).

Categories

filter

Synopsis

```
bjobs -plan
```

Description

To see the allocation plan, use the **bjobs -plan** option together with the `-l` option.

-prio

Displays the detailed absolute priority scheduling (APS) factor values for all pending jobs.

Categories

filter

Synopsis

```
bjobs -prio
```

Description

Displays the detailed priority factors and values for APS. Run **bqueues -r** for further details on fairshare user priority.

-psum

Displays a summarized version of reasons for pending jobs.

Categories

filter, format, state

Synopsis

```
bjobs -psum
```

Description

Displays the summarized number of jobs, hosts, and occurrences for each pending reason.

Note: By default **bjobs -psum** is equivalent to **bjobs -p -psum**.

By default, **bjobs** commands only access information about the submitting user's own jobs. Therefore, **bjobs -psum** presents a summary of only the submitting user's own pending jobs. If the `-u` option is specified for a specific user, user group, or all users (using the `all` keyword), pending jobs of the specific users are summarized. The **SECURE_JOB_INFO_LEVEL** parameter can define different access control levels for all users. The `-psum` option will only summarize the pending jobs that can be seen by the users specified with **SECURE_JOB_INFO_LEVEL**.

Conflicting options

Use only with the filter options that can return a list of pending jobs, including the following: `-app`, `-fwd`, `-G`, `-g`, `-J`, `-Jd`, `-Lp`, `-m`, `-P`, `-p`, `-p(0~3)`, `-pe`, `-pei`, `-pi`, `-q`, `-sla`, `-u`

Examples

```
bjobs -psum
```

Lists the top eligible and ineligible pending reasons in descending order by the number of jobs. If a host reason exists, further detailed host reasons are displayed in descending order by occurrences. Occurrence is a per-job per-host based number, counting the total times each job hits the reason on every single host.

```
#bjobs -p -psum
Pending reason summary: published Wed Mar 24 15:11:50 2016
Summarizing 100 pending jobs in cluster (cluster1):
    Individual host based reasons:                                    70 jobs
    Job's requirements for reserving resource (lic1) not satisfied:  20 jobs
    New job is waiting for scheduling:                                10 jobs

Individual host based reasons
    Load information unavailable:                                     160 occurrences
    Closed by LSF administrator:                                       80 occurrences
```

```
    Not specified in job submission:                                  40 occurrences
    Job requirements for reserving resource (mem) not satisfied:      40 occurrences
```

`bjobs -q short -u user1 -psum`

Displays summary of all pending job list submitted by user1 to the queue short.

`bjobs -p1 -psum`

Display summary of single key reasons (rather than all host pending reason).

`bjobs -p2 -psum`

Display summary of candidate host pending reasons (rather than non-candidate host pending reason).

`bjobs -p3 -psum`

Display summary of both candidate host pending reasons and non-candidate host pending reason.

-q

Displays jobs in the specified queue.

Categories

filter

Synopsis

`bjobs -q` *queue_name*

Description

The command **bqueues** returns a list of queues configured in the system, and information about the configurations of these queues.

In LSF multicluster capability, you cannot specify remote queues.

Examples

`bjobs -d -q short -m hostA -u user1`

Displays all the recently finished jobs submitted by `user1` to the queue `short`, and executed on the host `hostA`.

-r

Displays running jobs.

Categories

state

Synopsis

`bjobs -r`

-rusage

Displays jobs requesting the resources specified by the filter.

Categories

filter

Synopsis

`bjobs -rusage "`*resource_name*`[,`*resource_name*`,...]"`

Conflicting options

Can be used with other state or filter options (for example, `-p` or `-r`). Cannot be used with the `-A` option.

Description

Filters the results and shows only jobs that request the resource(s) specified.

If multiple resources are specified in the filter (separated by a comma with no spaces), the "AND" relationship applies.

If a job is submitted with multiple alternative resources in different sections (divided with "||"), each section will be considered separately for `rusage`. For example:

```
bsub -R "{rusage[resource1=1]} || {rusage[resource2=2]}"
```

Or

```
bsub -R "{rusage[resource1=1 || resource2=2]}"
```

In this case, `bjobs -rusage "resource1"` or `bjobs -rusage "resource2"` will return the job, but `bjobs -rusage "resource1,resource2"` will not return the job.

Examples

`bjobs -rusage "resource1"` Displays all jobs that are requesting resource1.

`bjobs -rusage "resource1" -p` Displays all pending jobs that are requesting resource1.

`bjobs -r -q normal -rusage "resource1,resourceA"` Displays all running jobs in the queue "normal" that are requesting resource1 and resourceA.

-s

Displays suspended jobs, together with the suspending reason that caused each job to become suspended.

Categories

state

Synopsis

```
bjobs -s
```

Description

The suspending reason may not remain the same while the job stays suspended. For example, a job may have been suspended due to the paging rate, but after the paging rate dropped another load index could prevent the job from being resumed. The suspending reason is updated according to the load index. The reasons could be as old as the time interval specified by **SBD_SLEEP_TIME** in `lsb.params`. The reasons shown may not reflect the current load situation.

Examples

```
bjobs -ps
```

Display only pending and suspended jobs.

-sla

Displays jobs belonging to the specified service class.

Categories

filter

Synopsis

`bjobs -sla` *service_class_name*

Description

bjobs also displays information about jobs assigned to a default SLA configured with ENABLE_DEFAULT_EGO_SLA in `lsb.params`.

Use **-sla** with **-g** to display job groups attached to a time-based service class. Once a job group is attached to a service class, all jobs submitted to that group are subject to the SLA.

Use **bsla** to display the configuration properties of service classes configured in `lsb.serviceclasses`, the default SLA configured in `lsb.params`, and dynamic information about the state of each service class.

Examples

```
bjobs -sla Sooke
```

Displays all jobs belonging to the service class Sooke.

-ss

Displays summary information for LSF Session Scheduler tasks.

Categories

filter

Synopsis

```
bjobs -ss
```

Conflicting options

Do not use with the following options: `-A`, `-aps`, `-fwd`, `-N`, `-W`, `-WL`, `-WF`, `-WP`.

Description

Displays summary information for LSF Session Scheduler tasks including the job ID, the owner, the job name (useful for job arrays), the total number of tasks, the state of pending, done, running, and exited session scheduler tasks.

`-ss` can only display the summary information for LSF Session Scheduler tasks when the job session has started . `-ss` cannot display the information while LSF Session Scheduler job is still pending.

The frequency of the updates of this information is based on the parameters **SSCHED_UPDATE_SUMMARY_INTERVAL** and **SSCHED_UPDATE_SUMMARY_BY_TASK**.

-sum

Displays summary information about unfinished jobs.

Categories

state, format

Synopsis

```
bjobs -sum
```

Description

bjobs -sum displays the count of job slots in the following states: running (RUN), system suspended (SSUSP), user suspended (USUSP), suspended while pending (PSUSP), pending (PEND), forwarded to remote clusters and pending (FWD_PEND), and UNKNOWN.

bjobs -sum displays the job slot count only for the user's own jobs.

Use **-sum** with other options (like **-m**, **-P**, **-q**, and **-u**) to filter the results. For example, **bjobs -sum -u user1** displays job slot counts just for user `user1`.

Examples

```
% bjobs -sum
RUN         SSUSP       USUSP      UNKNOWN    PEND       FWD_PEND  PSUSP
123         456         789        5          5          3         3
```

To filter the **-sum** results to display job slot counts just for user `user1`, run **bjobs -sum -u user1**:

```
% bjobs -sum -u user1
RUN         SSUSP       USUSP      UNKNOWN    PEND       FWD_PEND  PSUSP
20          10          10         0           5          0        2
```

-U

Displays jobs that are associated with the specified advance reservation.

Categories

filter

Synopsis

`bjobs -U` *adv_reservation_id*

Description

To view all jobs that are associated with the specified advance reservation, run the `-U` option with `0` as the job ID.

To see if a single job is associated with an advance reservation, run the `-U` option with one specified job ID. The output displays the job information only if the job is associated with the specified advance reservation. Otherwise, the output displays an error message.

To see which jobs in a specified list are associated with an advance reservation, run the `-U` and `-l` options together with multiple job IDs. The output displays detailed job information only if the jobs are associated with the specified advanced reservation. Otherwise, the output displays an error message for each job that is not associated with the advanced reservation.

Note: To view a list of job IDs, you must use the `-U` and `-l` options together. If you use the `-U` option without `-l` for multiple job IDs, the output displays all jobs regardless of whether they are associated with the advance reservation unless the job IDs are not valid.

Examples

Displays all jobs that are associated with the reservation ID `user1#2`:

```
bjobs -U user1#2 0
```

Displays one job with ID 101 if it is associated with the reservation ID `user1#2`:

```
bjobs -U user1#2 101
```

Displays any jobs from a specified list (with IDs 101, 102, 103, and 104) that are associated with the reservation ID `user1#2`:

```
bjobs -U user1#2 -l 101 102 103 104
```

-UF

Displays unformatted job detail information.

Categories

format

Synopsis

`bjobs -UF`

Description

This makes it easy to write scripts for parsing keywords on **bjobs**. The results of this option have no wide control for the output. Each line starts from the beginning of the line. Information for **SCHEDULING PARAMETERS** and **PENDING REASONS** remain formatted. The resource usage message lines ending without any separator have a semicolon added to separate their different parts. The first line and all lines starting with the time stamp are displayed unformatted in a single line. There is no line length and format control.

Examples

```
% bjobs -UF
Job <1>, User <lsfuser>, Project <default>, Status <RUN>, Queue <normal>, Command <./pi_css5
10000000>, Share group charged </lsfuser>
Tue May  6 15:45:10: Submitted from host <hostA>, CWD </home/lsfuser>;
Tue May  6 15:45:11: Started on <hostB>, Execution Home </home/lsfuser>, Execution CWD </home/
lsfuser>;

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

 RESOURCE REQUIREMENT DETAILS:
 Combined: select[type == local] order[r15s:pg]
 Effective: select[type == local] order[r15s:pg]
```

-u

Displays jobs that were submitted by the specified users or user groups.

Categories

filter

Synopsis

`bjobs -u` *user_name* ... | `-u` *user_group* ... | `-u all`

Conflicting options

Do not use with the `-G` option.

Description

The keyword all specifies all users. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

Examples

```
bjobs -u all -a
```

Displays all jobs of all users.

```
bjobs -d -q short -m hostA -u user1
```

Displays all the recently finished jobs submitted by `user1` to the queue `short`, and executed on the host `hostA`.

-W

Provides resource usage information for: PROJ_NAME, CPU_USED, MEM, SWAP, PIDS, START_TIME, FINISH_TIME.

Categories

format

Synopsis

```
bjobs -W
```

Description

Displays resource information for jobs that belong to you only if you are not logged in as an administrator.

-WF

Displays an estimated finish time for running or pending jobs. For done or exited jobs, displays the actual finish time.

Categories

format

Synopsis

```
bjobs -WF
```

Output

The output for the `-WF`, `-WL`, and `-WP` options are in the following format:

`hours:minutes status`

where *status* is one of the following:

- X: The real run time has exceeded the estimated run time configured in the application profile (**RUNTIME** parameter in `lsb.applications`) or at the job level (**bsub -We** option).
- L: A run limit exists but the job does not have an estimated run time.
- E: An estimated run time exists and has not been exceeded.

-WL

Displays the estimated remaining run time of jobs.

Categories

format

Synopsis

```
bjobs -WL
```

Output

The output for the `-WF`, `-WL`, and `-WP` options are in the following format:

hours`:`*minutes status*

where *status* is one of the following:

- X: The real run time has exceeded the estimated run time configured in the application profile (**RUNTIME** parameter in `lsb.applications`) or at the job level (**bsub -We** option).
- L: A run limit exists but the job does not have an estimated run time.
- E: An estimated run time exists and has not been exceeded.

-WP

Displays the current estimated completion percentage of jobs.

Categories

format

Synopsis

```
bjobs -WP
```

Output

The output for the `-WF`, `-WL`, and `-WP` options are in the following format:

hours`:`*minutes status*

where *status* is one of the following:

- X: The real run time has exceeded the estimated run time configured in the application profile (**RUNTIME** parameter in `lsb.applications`) or at the job level (**bsub -We** option).
- L: A run limit exists but the job does not have an estimated run time.
- E: An estimated run time exists and has not been exceeded.

-w

Wide format. Displays job information without truncating fields.

Categories

format

Synopsis

```
bjobs -w
```

-X

Displays uncondensed output for host groups and compute units.

Categories

format

Synopsis

```
bjobs -X
```

Examples

```
bjobs -X 101 102 203 509
```

Display jobs with job ID 101, 102, 203, and 509 as uncondensed output even if these jobs belong to hosts in condensed groups.

-x

Displays unfinished jobs that have triggered a job exception (overrun, underrun, idle, runtime_est_exceeded).

Categories

state

Synopsis

```
bjobs -x
```

Description

Use with the `-l` option to show the actual exception status. Use with `-a` to display all jobs that have triggered a job exception.

job_id

Specifies the jobs or job arrays that **bjobs** displays.

Synopsis

`bjobs` [*options*] [*job_id* | "*job_id*[*index_list*]" ...]

Description

If you use `-A`, specify job array IDs without the index list.

In LSF multicluster capability job forwarding mode, you can use the local job ID and cluster name to retrieve the job details from the remote cluster. The query syntax is:

`bjobs` *submission_job_id*`@`*submission_cluster_name*

For job arrays, the query syntax is:

`bjobs "`*submission_job_id*`[`*index*`]"@`*submission_cluster_name*

The advantage of using **submission_job_id@submission_cluster_name** instead of **bjobs -l job_id** is that you can use **submission_job_id@submission_cluster_name** as an alias to query a local job in the execution cluster without knowing the local job ID in the execution cluster. The **bjobs** output is identical no matter which job ID you use (local job ID or *submission_job_id*`@`*submission_cluster_name*).

You can use **bjobs 0** to find all jobs in your local cluster, but **bjobs 0@submission_cluster_name** is not supported.

Examples

```
bjobs 101 102 203 509
```

Display jobs with job_ID 101, 102, 203, and 509.

```
bjobs -X 101 102 203 509
```

Display jobs with job ID 101, 102, 203, and 509 as uncondensed output even if these jobs belong to hosts in condensed groups.

-h

Displays a description of the specified category, command option, or sub-option to `stderr` and exits.

Synopsis

`bjobs -h[elp] [`*category* ...`] [`*option* ...`]`

Description

You can abbreviate the `-help` option to `-h`.

Run **bjobs -h** (or **bjobs -help**) without a command option or category name to display the **bjobs** command description.

Examples

```
bjobs -h filter
```

Displays a description of the `filter` category and the **bjobs** command options belonging to this category.

```
bjobs -h -o
```

Displays a detailed description of the **bjobs -o** option.

-V

Prints LSF release version to `stderr` and exits.

Synopsis

```
bjobs -V
```

Conflicting options

Do not use with any other option except `-h` (**bjobs -h -V**).

Description

By default, displays information about your own pending, running, and suspended jobs.

bjobs displays output for condensed host groups and compute units. These host groups and compute units are defined by CONDENSE in the `HostGroup` or `ComputeUnit` section of `lsb.hosts`. These groups are displayed as a single entry with the name as defined by GROUP_NAME or NAME in `lsb.hosts`. The `-l` and `-X` options display uncondensed output.

If you defined the parameter **LSB_SHORT_HOSTLIST=1** in the `lsf.conf` file, parallel jobs running in the same condensed host group or compute unit are displayed as an abbreviated list.

For resizable jobs, **bjobs** displays the autoresizable attribute and the resize notification command.

To display older historical information, use **bhist**.

Output: Default Display

Pending jobs are displayed in the order in which they are considered for dispatch. Jobs in higher priority queues are displayed before those in lower priority queues. Pending jobs in the same priority queues are displayed in the order in which they were submitted but this order can be changed by using the commands **btop** or **bbot**. If more than one job is dispatched to a host, the jobs on that host are listed in the order in which they are considered for scheduling on this host by their queue priorities and dispatch times. Finished jobs are displayed in the order in which they were completed.

A listing of jobs is displayed with the following fields:

JOBID
The job ID that LSF assigned to the job.

USER
The user who submitted the job.

STAT
The current status of the job (see JOB STATUS below).

QUEUE
The name of the job queue to which the job belongs. If the queue to which the job belongs has been removed from the configuration, the queue name is displayed as `lost_and_found`. Use **bhist** to get the original queue name. Jobs in the `lost_and_found` queue remain pending until they are switched with the **bswitch** command into another queue.

In a LSF multicluster capability resource leasing environment, jobs scheduled by the consumer cluster display the remote queue name in the format *queue_name*@*cluster_name*. By default, this field truncates at 10 characters, so you might not see the cluster name unless you use `-w` or `-l`.

FROM_HOST
The name of the host from which the job was submitted.

With the LSF multicluster capability, if the host is in a remote cluster, the cluster name and remote job ID are appended to the host name, in the format *host_name*@*cluster_name*:*job_ID*. By default, this field truncates at 11 characters; you might not see the cluster name and job ID unless you use `-w` or `-l`.

EXEC_HOST
The name of one or more hosts on which the job is executing (this field is empty if the job has not been dispatched). If the host on which the job is running has been removed from the configuration, the host name is displayed as `lost_and_found`. Use **bhist** to get the original host name.

If the host is part of a condensed host group or compute unit, the host name is displayed as the name of the condensed group.

If you configure a host to belong to more than one condensed host groups using wildcards, **bjobs** can display any of the host groups as execution host name.

JOB_NAME
The job name assigned by the user, or the command string assigned by default at job submission with **bsub**. If the job name is too long to fit in this field, then only the latter part of the job name is displayed.

The displayed job name or job command can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

SUBMIT_TIME
The submission time of the job.

Output: Long format (-l)

The `-l` option displays a long format listing with the following additional fields:

Job
The job ID that LSF assigned to the job.

User
The ID of the user who submitted the job.

Project
The project the job was submitted from.

Application Profile
The application profile the job was submitted to.

Command
The job command.

CWD
The current working directory on the submission host.

Data requirement requested
Indicates that the job has data requirements.

Execution CWD
The actual CWD used when job runs.

Host file

The path to a user-specified host file used when submitting or modifying a job.

Initial checkpoint period
The initial checkpoint period specified at the job level, by **bsub -k**, or in an application profile with CHKPNT_INITPERIOD.

Checkpoint period
The checkpoint period specified at the job level, by **bsub -k**, in the queue with CHKPNT, or in an application profile with CHKPNT_PERIOD.

Checkpoint directory
The checkpoint directory specified at the job level, by **bsub -k**, in the queue with CHKPNT, or in an application profile with CHKPNT_DIR.

Migration threshold
The migration threshold specified at the job level, by **bsub -mig**.

Post-execute Command
The post-execution command specified at the job-level, by **bsub -Ep**.

PENDING REASONS
The reason the job is in the PEND or PSUSP state. The names of the hosts associated with each reason are displayed when both `-p` and `-l` options are specified.

SUSPENDING REASONS
The reason the job is in the USUSP or SSUSP state.

loadSched
The load scheduling thresholds for the job.

loadStop
The load suspending thresholds for the job.

JOB STATUS
Possible values for the status of a job include:

PEND

The job is pending. That is, it has not yet been started.

PROV
The job has been dispatched to a power-saved host that is waking up. Before the job can be sent to the sbatchd, it is in a PROV state.

PSUSP
The job has been suspended, either by its owner or the LSF administrator, while pending.

RUN
The job is currently running.

USUSP
The job has been suspended, either by its owner or the LSF administrator, while running.

SSUSP
The job has been suspended by LSF. The following are examples of why LSF suspended the job:

- The load conditions on the execution host or hosts have exceeded a threshold according to the `loadStop` vector defined for the host or queue.
- The run window of the job's queue is closed. See **bqueues(1)**, **bhosts(1)**, and `lsb.queues(5)`.

DONE
The job has terminated with status of 0.

EXIT
The job has terminated with a non-zero status – it may have been aborted due to an error in its execution, or killed by its owner or the LSF administrator.

For example, exit code 131 means that the job exceeded a configured resource usage limit and LSF killed the job.

UNKWN
mbatchd has lost contact with the sbatchd on the host on which the job runs.

WAIT
For jobs submitted to a chunk job queue, members of a chunk job that are waiting to run.

ZOMBI
A job becomes ZOMBI if:

- A non-rerunnable job is killed by **bkill** while the sbatchd on the execution host is unreachable and the job is shown as UNKWN.
- After the execution host becomes available, LSF tries to kill the ZOMBI job. Upon successful termination of the ZOMBI job, the job's status is changed to EXIT.

  With the LSF multicluster capability, when a job running on a remote execution cluster becomes a ZOMBI job, the execution cluster treats the job the same way as local ZOMBI jobs. In addition, it notifies the submission cluster that the job is in ZOMBI state and the submission cluster requeues the job.

RUNTIME
Estimated run time for the job, specified by **bsub -We** or **bmod -We**, -We+, -Wep.

The following information is displayed when running **bjobs -WL**, **-WF**, or **-WP**.

TIME_LEFT
The estimated run time that the job has remaining. Along with the time if applicable, one of the following symbols may also display.

- E: The job has an estimated run time that has not been exceeded.
- L: The job has a hard run time limit specified but either has no estimated run time or the estimated run time is more than the hard run time limit.
- X: The job has exceeded its estimated run time and the time displayed is the time remaining until the job reaches its hard run time limit.
- A dash indicates that the job has no estimated run time and no run limit, or that it has exceeded its run time but does not have a hard limit and therefore runs until completion.

If there is less than a minute remaining, 0:0 displays.

FINISH_TIME
The estimated finish time of the job. For done/exited jobs, this is the actual finish time. For running jobs, the finish time is the start time plus the estimated run time (where set and not exceeded) or the start time plus the hard run limit.

- E: The job has an estimated run time that has not been exceeded.
- L: The job has a hard run time limit specified but either has no estimated run time or the estimated run time is more than the hard run time limit.
- X: The job has exceeded its estimated run time and had no hard run time limit set. The finish time displayed is the estimated run time remaining plus the start time.
- A dash indicates that the pending, suspended, or job with no run limit has no estimated finish time.

%COMPLETE
The estimated completion percentage of the job.

- E: The job has an estimated run time that has not been exceeded.
- L: The job has a hard run time limit specified but either has no estimated run time or the estimated run time is more than the hard run time limit.
- X: The job has exceeded its estimated run time and had no hard run time limit set.
- A dash indicates that the jobs is pending, or that it is running or suspended, but has no run time limit specified.

Note: For jobs in the state UNKNOWN, the job run time estimate is based on internal counting by the job's **mbatchd**.

RESOURCE USAGE

For the LSF multicluster capability job forwarding model, this information is not shown if the LSF multicluster capability resource usage updating is disabled. Use **LSF_HPC_EXTENSIONS="HOST_RUSAGE"** in `lsf.conf` to specify host-based resource usage.

The values for the current usage of a job include:

HOST
For host-based resource usage, specifies the host.

CPU time
Cumulative total CPU time in seconds of all processes in a job. For host-based resource usage, the cumulative total CPU time in seconds of all processes in a job running on a host.

IDLE_FACTOR
Job idle information (CPU time/runtime) if JOB_IDLE is configured in the queue, and the job has triggered an idle exception.

MEM
Total resident memory usage of all processes in a job. For host-based resource usage, the total resident memory usage of all processes in a job running on a host. The sum of host-based rusage may not equal the total job rusage, since total job rusage is the maximum historical value.

Memory usage unit is scaled automatically based on the value. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify the smallest unit for display (KB, MB, GB, TB, PB, or EB).

SWAP
Total virtual memory and swap usage of all processes in a job. For host-based resource usage, the total virtual memory usage of all processes in a job running on a host. The sum of host-based usage may not equal the total job usage, since total job usage is the maximum historical value.

Swap usage unit is scaled automatically based on the value. Use the **LSF_UNIT_FOR_LIMITS** in the `lsf.conf` file to specify the smallest unit for display (KB, MB, GB, TB, PB, or EB).

By default, LSF collects both memory and swap usage through PIM:

- If the **EGO_PIM_SWAP_REPORT=n** parameter is set in the `lsf.conf` file (this is the default), swap usage is virtual memory (VSZ) of the entire job process.
- If the **EGO_PIM_SWAP_REPORT=y** parameter is set in the `lsf.conf` file, the resident set size (RSS) is subtracted from the virtual memory usage. RSS is the portion of memory occupied by a process that is held in main memory. Swap usage is collected as the *VSZ - RSS*.

If memory enforcement through the Linux cgroup memory subsystem is enabled with the **LSF_LINUX_CGROUP_ACCT=y** parameter in the `lsf.conf` file, LSF uses the cgroup memory subsystem to collect memory and swap usage of all processes in a job.

NTHREAD
Number of currently active threads of a job.

PGID
Currently active process group ID in a job. For host-based resource usage, the currently active process group ID in a job running on a host.

PIDs
Currently active processes in a job. For host-based resource usage, the currently active active processes in a job running on a host.

RESOURCE LIMITS
The hard resource usage limits that are imposed on the jobs in the queue (see getrlimit(2) and `lsb.queues`(5)). These limits are imposed on a per-job and a per-process basis.

The possible per-job resource usage limits are:

- CPULIMIT
- TASKLIMIT
- MEMLIMIT
- SWAPLIMIT
- PROCESSLIMIT
- THREADLIMIT
- OPENFILELIMIT
- HOSTLIMIT_PER_JOB

The possible UNIX per-process resource usage limits are:

- RUNLIMIT
- FILELIMIT
- DATALIMIT
- STACKLIMIT
- CORELIMIT

If a job submitted to the queue has any of these limits specified (see **bsub**(1)), then the lower of the corresponding job limits and queue limits are used for the job.

If no resource limit is specified, the resource is assumed to be unlimited. User shell limits that are unlimited are not displayed.

EXCEPTION STATUS
Possible values for the exception status of a job include:

idle
The job is consuming less CPU time than expected. The job idle factor (CPU time/runtime) is less than the configured JOB_IDLE threshold for the queue and a job exception has been triggered.

overrun
The job is running longer than the number of minutes specified by the JOB_OVERRUN threshold for the queue and a job exception has been triggered.

underrun
The job finished sooner than the number of minutes specified by the JOB_UNDERRUN threshold for the queue and a job exception has been triggered.

Requested resources
Shows all the resource requirement strings you specified in the **bsub** command.

Execution rusage
This is shown if the combined RES_REQ has an rusage OR || construct. The chosen alternative will be denoted here.

Synchronous Execution
Job was submitted with the `-K` option. LSF submits the job and waits for the job to complete.

JOB_DESCRIPTION
The job description assigned by the user. This field is omitted if no job description has been assigned.

The displayed job description can contain up to 4094 characters.

MEMORY USAGE
Displays peak memory usage and average memory usage. For example:

```
MEMORY USAGE:

MAX MEM:11 Mbytes; AVG MEM:6 Mbytes
```

You can adjust rusage accordingly next time for the same job submission if consumed memory is larger or smaller than current rusage.

RESOURCE REQUIREMENT DETAILS
Displays the configured level of resource requirement details. The **BJOBS_RES_REQ_DISPLAY** parameter in `lsb.params` controls the level of detail that this column displays, which can be as follows:

- none - no resource requirements are displayed (this column is not displayed in the **-l** output).
- brief - displays the combined and effective resource requirements.
- full - displays the job, app, queue, combined and effective resource requirements.

Requested Network
Displays network resource information for IBM Parallel Edition (PE) jobs submitted with the `bsub -network` option. It does not display network resource information from the **NETWORK_REQ** parameter in `lsb.queues` or `lsb.applications`.

For example:

```
bjobs -l
Job <2106>, User <user1>;, Project <default>;, Status <RUN>;, Queue <normal>,
                     Command <my_pe_job>
Fri Jun  1 20:44:42: Submitted from host <hostA>, CWD <$HOME>, Requested Network
                      <protocol=mpi: mode=US: type=sn_all: instance=1: usage=dedicated>
```

If `mode=IP` is specified for the PE job, `instance` is not displayed.

DATA REQUIREMENTS
When you use `-data`, displays a list of requested files for jobs with data requirements.

Output: Forwarded job information

The `-fwd` option filters output to display information on forwarded jobs in the LSF multicluster capability job forwarding mode. The following additional fields are displayed:

CLUSTER
The name of the cluster to which the job was forwarded.

FORWARD_TIME
The time that the job was forwarded.

Output: Job array summary information

Use `-A` to display summary information about job arrays. The following fields are displayed:

JOBID

Job ID of the job array.

ARRAY_SPEC
Array specification in the format of *name*[*index*]. The array specification may be truncated, use `-w` option together with `-A` to show the full array specification.

OWNER
Owner of the job array.

NJOBS
Number of jobs in the job array.

PEND
Number of pending jobs of the job array.

RUN

Number of running jobs of the job array.

DONE

Number of successfully completed jobs of the job array.

EXIT

Number of unsuccessfully completed jobs of the job array.

SSUSP

Number of LSF system suspended jobs of the job array.

USUSP

Number of user suspended jobs of the job array.

PSUSP

Number of held jobs of the job array.

Output: LSF Session Scheduler job summary information

JOBID

Job ID of the Session Scheduler job.

OWNER

Owner of the Session Scheduler job.

JOB_NAME

The job name assigned by the user, or the command string assigned by default at job submission with **bsub**. If the job name is too long to fit in this field, then only the latter part of the job name is displayed.

The displayed job name or job command can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

NTASKS

The total number of tasks for this Session Scheduler job.

PEND

Number of pending tasks of the Session Scheduler job.

RUN

Number of running tasks of the Session Scheduler job.

DONE

Number of successfully completed tasks of the Session Scheduler job.

EXIT

Number of unsuccessfully completed tasks of the Session Scheduler job.

Output: Unfinished job summary information

Use `-sum` to display summary information about unfinished jobs. The count of job slots for the following job states is displayed:

RUN

The job is running.

SSUSP

The job has been suspended by LSF.

USUSP

The job has been suspended, either by its owner or the LSF administrator, while running.

UNKNOWN

mbatchd has lost contact with the sbatchd on the host where the job was running.

PEND
The job is pending, which may include PSUSP and chunk job WAIT. When -sum is used with -p in the LSF multicluster capability, WAIT jobs are not counted as PEND or FWD_PEND. When -sum is used with -r, WAIT jobs are counted as PEND or FWD_PEND.

FWD_PEND
The job is pending and forwarded to a remote cluster. The job has not yet started in the remote cluster.

Output: Affinity resource requirements information (-l -aff)

Use -l -aff to display information about CPU and memory affinity resource requirements for job tasks. A table with the heading AFFINITY is displayed containing the detailed affinity information for each task, one line for each allocated processor unit. CPU binding and memory binding information are shown in separate columns in the display.

HOST

The host the task is running on

TYPE

Requested processor unit type for CPU binding. One of numa, socket, core, or thread.

LEVEL

Requested processor unit binding level for CPU binding. One of numa, socket, core, or thread. If no CPU binding level is requested, a dash (-) is displayed.

EXCL

Requested processor unit binding level for exclusive CPU binding. One of numa, socket, or core. If no exclusive binding level is requested, a dash (-) is displayed.

IDS

List of physical or logical IDs of the CPU allocation for the task.

The list consists of a set of paths, represented as a sequence integers separated by slash characters (/), through the topology tree of the host. Each path identifies a unique processing unit allocated to the task. For example, a string of the form 3/0/5/12 represents an allocation to thread 12 in core 5 of socket 0 in NUMA node 3. A string of the form 2/1/4represents an allocation to core 4 of socket 1 in NUMA node 2. The integers correspond to the node ID numbers displayed in the topology tree from **bhosts -aff**.

POL

Requested memory binding policy. Eitherlocal or pref. If no memory binding is requested, a dash (-) is displayed.

NUMA

ID of the NUMA node that the task memory is bound to. If no memory binding is requested, a dash (-) is displayed.

SIZE

Amount of memory allocated for the task on the NUMA node.

For example the following job starts 6 tasks with the following affinity resource requirements:

```
bsub -n 6 -R"span[hosts=1] rusage[mem=100]affinity[core(1,same=socket,
exclusive=(socket,injob)):cpubind=socket:membind=localonly:distribute=pack]" myjob
Job <6> is submitted to default queue <normal>.
```

```
bjobs -l -aff 6

Job <6>, User <user1>, Project <default>, Status <RUN>, Queue <normal>, Comman
                     d <myjob1>
```

```
Thu Feb 14 14:13:46: Submitted from host <hostA>, CWD <$HOME>, 6 Task(s),
                     Requested Resources <span[hosts=1] rusage[mem=10
                     0]affinity[core(1,same=socket,exclusive=(socket,injob)):cp
                     ubind=socket:membind=localonly:distribute=pack]>;
Thu Feb 14 14:15:07: Started 6 Task(s) on Hosts <hostA> <hostA> <hostA> <hostA>
                     <hostA> <hostA>, Allocated 6 Slot(s) on Hosts <hostA>
                     <hostA> <hostA> <hostA> <hostA> <hostA>, Execution Home
                     </home/user1>, Execution CWD </home/user1>;

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

 RESOURCE REQUIREMENT DETAILS:
 Combined: select[type == local] order[r15s:pg] rusage[mem=100.00] span[hosts=1
                     ] affinity[core(1,same=socket,exclusive=(socket,injob))*1:
                     cpubind=socket:membind=localonly:distribute=pack]
 Effective: select[type == local] order[r15s:pg] rusage[mem=100.00] span[hosts=
                     1] affinity[core(1,same=socket,exclusive=(socket,injob))*1
                     :cpubind=socket:membind=localonly:distribute=pack]

 AFFINITY:
                     CPU BINDING                          MEMORY BINDING
                     -----------------------              --------------------
 HOST                TYPE   LEVEL  EXCL   IDS            POL   NUMA SIZE
 hostA               core   socket socket /0/0/0         local 0    16.7MB
 hostA               core   socket socket /0/1/0         local 0    16.7MB
 hostA               core   socket socket /0/2/0         local 0    16.7MB
 hostA               core   socket socket /0/3/0         local 0    16.7MB
 hostA               core   socket socket /0/4/0         local 0    16.7MB
 hostA               core   socket socket /0/5/0         local 0    16.7MB
```

Output: Data requirements information (-l -data)

Use `-l -data` to display detailed information about jobs with data requirements. The heading DATA REQUIREMENTS is displayed followed by a list of the files requested by the job.

For example:

```
bjobs -l -data 1962
Job <1962>, User <user1>, Project <default>, Status <PEND>, Queue
        <normal>,Command <my_data_job>
Fri Sep 20 16:31:17: Submitted from host <hb05b10>, CWD
        <$HOME/source/user1/work>, Data requirement requested;
 PENDING REASONS:
 Job is waiting for its data requirement to be satisfied;

 SCHEDULING PARAMETERS:
           r15s  r1m  r15m  ut  pg  io  ls  it  tmp  swp  mem
 loadSched   -     -   -     -       -     -    -     -     -      -      -
 loadStop    -     -   -     -       -     -    -     -     -      -      -

 RESOURCE REQUIREMENT DETAILS:
 Combined: select[type == local] order[r15s:pg]
 Effective: -

 DATA REQUIREMENTS:
 FILE: hostA:/home/user1/data2
 SIZE: 40 MB
 MODIFIED: Thu Aug 14 17:01:57

 FILE: hostA:/home/user1/data3
 SIZE: 40 MB
 MODIFIED: Fri Aug 15 16:32:45

 FILE: hostA:/home/user1/data4
 SIZE: 500 MB
 MODIFIED: Mon Apr 14 17:15:56
```

See also

bsub, **bkill**, **bhosts**, **bmgroup**, **bclusters**, **bqueues**, **bhist**, **bresume**, **bsla**, **bstop**, lsb.params, lsb.serviceclasses, **mbatchd**

Chapter 21. bkill

Sends signals to kill, suspend, or resume unfinished jobs

Synopsis

`bkill` [`-l`] [`-app` *application_profile_name*] [`-g` *job_group_name*] [`-sla` *service_class_name*] [`-J` *job_name*] [`-m` *host_name* | `-m` *host_group*] [`-q` *queue_name*] [`-r` | `-s` *signal_value* | *signal_name*] [`-u` *user_name* | `-u` *user_group* | `-u all`] [*job_ID* ... | `0` | "*job_ID*[*index*]" ...]

`bkill` [`-l`] [`-b`] [`-app` *application_profile_name*] [`-g` *job_group_name*] [`-sla` *service_class_name*] [`-J` *job_name*] [`-m` *host_name* | `-m` *host_group*] [`-q` *queue_name*] [`-u` *user_name* | `-u` *user_group* | `-u all`] [`-C` *kill_reason*] [*job_ID* ... | `0` | "*job_ID*[*index*]" ...]

`bkill` [`-h` | `-V`]

Description

By default, sends a set of signals to kill the specified jobs. On UNIX, `SIGINT` and `SIGTERM` signals are sent to give the job a chance to clean up before termination, then the `SIGKILL` signal is sent to kill the job. The time interval between sending each signal is defined by the **JOB_TERMINATE_INTERVAL** parameter in the `lsb.params` file.

By default, kills the last job that was submitted by the user who ran the command. You must specify a job ID or the `-app`, `-g`, `-J`, `-m`, `-u`, or `-q` option. If you specify the `-app`, `-g`, `-J`, `-m`, `-u`, or `-q` option without a job ID, the **bkill** command kills the last job that was submitted by the user who ran the command. Specify job ID **0** to kill multiple jobs.

On Windows, job control messages replace the `SIGINT` and `SIGTERM` signals, but only customized applications can process them. The `TerminateProcess()` system call is sent to kill the job.

The **bkill** command sends the signals `INT`, `TERM`, and `KILL` in sequence. The exit code that is returned when a dispatched job is killed with the **bkill** command depends on which signal killed the job.

If **PRIVILEGED_USER_FORCE_BKILL=y** parameter is set in the `lsb.params` file, only root and LSF administrators can run the **bkill -r** command. The `-r` option is ignored for other users.

Users can operate only on their own jobs. Only root and LSF administrators can operate on jobs that are submitted by other users.

If a signal request fails to reach the job execution host, LSF tries the operation later when the host becomes reachable. LSF retries the most recent signal request.

If a job is running in a queue with `CHUNK_JOB_SIZE` set, the **bkill** command has the following results, depending on job state:

PEND
Job is removed from chunk (`NJOBS -1`, `PEND -1`)

RUN
All jobs in the chunk are suspended (`NRUN -1`, `NSUSP +1`)

USUSP
Job finishes, next job in the chunk starts if one exists (`NJOBS -1`, `PEND -1`, `SUSP -1`, `RUN +1`)

WAIT
Job finishes (`NJOBS-1`, `PEND -1`)

If the job cannot be killed, use the **bkill -r** command to remove the job from the LSF system without waiting for the job to terminate, and free the resources of the job.

Options

0
Kills all the jobs that satisfy other options (-app. -g, -m, -q, -u, and -J options).

-b
Kills large numbers of jobs as soon as possible. Local pending jobs are killed immediately and cleaned up as soon as possible, ignoring the time interval specified by the **CLEAN_PERIOD** parameter in the lsb.params file. Jobs that are killed in this manner are not logged to the lsb.acct file.

Other jobs, such as running jobs, are killed as soon as possible and cleaned up normally.

If the -b option is used with the 0 subcommand, the **bkill** command kills all applicable jobs and silently skips the jobs that cannot be killed.

```
bkill -b 0
Operation is in progress
```

The -b option is ignored if used with the -r or -s options.

-l
Displays the signal names that are supported by the **bkill** command. The supported signals are a subset of signals that are supported by the /bin/kill command and is operating system-dependent.

-r
Removes a job from the LSF system without waiting for the job to terminate in the operating system.

If the **PRIVILEGED_USER_FORCE_BKILL=y** parameter is set in the lsb.params file, only root and LSF administrators can run the **bkill -r** command. The -r option is ignored for other users.

Sends the same series of signals as the **bkill** command without the -r option, except that the job is removed from the system immediately. If the job is in UNKNWN state, the **bkill -r** command marks the job as ZOMBIE state. The **bkill -r** command changes jobs in ZOMBIE state to EXIT state. The job resources that LSF monitors are released as soon as LSF receives the first signal.

Use the **bkill -r** command only on jobs that cannot be killed in the operating system, or on jobs that cannot be otherwise removed by using the **bkill** command.

The -r option cannot be used with the -s option.

-app *application_profile_name*
Operates only on jobs that are associated with the specified application profile. You must specify an existing application profile. If *job_ID* or 0 is not specified, only the most recently submitted qualifying job is operated on.

-C *kill_reason*
Gives the user the option to add a reason as to why the job is being killed. The length of the reason is limited to 4095 characters.

Note: This option cannot be used in combination with the -s option.

-g *job_group_name*
Operates only on jobs in the job group that is specified by *job_group_name*.

Use the -g option with the -sla option to kill jobs in job groups that are attached to a service class.

The **bkill** command does not kill jobs in lower-level job groups in the path. For example, jobs are attached to job groups /risk_group and /risk_group/consolidate:

```
bsub -g /risk_group  myjob
Job <115> is submitted to default queue <normal>.
```

```
bsub -g /risk_group/consolidate myjob2
Job <116> is submitted to default queue <normal>.
```

The following **bkill** command kills only jobs in the /risk_group job group, not the subgroup /risk_group/consolidate:

```
bkill -g /risk_group 0
Job <115> is being terminated
```

```
bkill -g /risk_group/consolidate 0
Job <116> is being terminated
```

-J ***job_name***
Operates only on jobs with the specified job name. The -J option is ignored if a job ID other than 0 is specified in the *job_ID* option.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern job* returns jobA and jobarray[1]. The *AAA*[1] pattern returns the first element in job arrays with names that contain AAA. However, the pattern job1[*] does not return anything since the wildcard is within the array index.

-m ***host_name*** **| -m** ***host_group***
Operates only on jobs that are dispatched to the specified host or host group.

If *job_ID* is not specified, only the most recently submitted qualifying job is operated on. The -m option is ignored if a job ID other than 0 is specified in the *job_ID* option. Use the **bhosts** and **bmgroup** commands to see information about hosts and host groups.

-q ***queue_name***
Operates only on jobs in the specified queue.

If *job_ID* is not specified, only the most recently submitted qualifying job is operated on.

The -q option is ignored if a job ID other than 0 is specified in the *job_ID* option.

Use the **bqueues**command to see information about queues.

-s ***signal_value*** **|** ***signal_name***
Sends the specified signal to specified jobs. You can specify either a name, stripped of the SIG prefix (such as KILL), or a number (such as 9).

Eligible UNIX signal names are listed by the **bkill -l** command.

The -s option cannot be used with the -r option.

Use the **bkill -s** command to suspend and resume jobs by using the appropriate signal instead of using the **bstop** or **bresume** command. Sending the SIGCONT signal is the same as using the **bresume** command.

Sending the SIGSTOP signal to sequential jobs or the SIGTSTP signal to parallel jobs is the same as using the **bstop** command.

You cannot suspend a job that is already suspended, or resume a job that is not suspended. Using the SIGSTOP or SIGTSTP signal on a job that is in the USUSP state has no effect. Using the SIGCONT signal on a job that is not in either the PSUSP or the USUSP state has no effect. Use the **bjobs** command to see information about job states.

Limited Windows signals are supported:

- bkill -s 7 or bkill SIGKILL to terminate a job
- bkill -s 16 or bkill SIGSTOP to suspend a job
- bkill -s 15 to resume a job

-sla ***service_class_name***
Operates on jobs that belong to the specified service class.

If a job ID is not specified, only the most recently submitted job is operated on.

Use the `-sla` option with the `-g` option to kill jobs in job groups that are attached to a service class.

The `-sla` option is ignored if a job ID other than 0 is specified in the *job_ID* option.

Use the **bsla** command to display the configuration properties of service classes that are configured in the `lsb.serviceclasses` file. The **bsla** command also shows the default SLA configured with the **ENABLE_DEFAULT_EGO_SLA** parameter in the `lsb.params` file, and dynamic information about the state of each service class.

-u *user_name* | -u *user_group* | -u all
Operates only on jobs that are submitted by the specified user or user group, or by all users if the reserved user name `all` is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

If a job ID is not specified, only the most recently submitted qualifying job is operated on. The `-u` option is ignored if a job ID other than `0` is specified in the *job_ID* option.

***job_ID* ... | 0 | "*job_ID*[*index*]" ...**
Operates only on jobs that are specified by *job_ID* or "*job_ID*[*index*]", where "*job_ID*[*index*]" specifies selected job array elements. Use the **bjobs** command to see the array elements for the job. For job arrays, quotation marks must enclose the job ID and index, and index must be enclosed in square brackets.

Kill an entire job array by specifying the job array ID instead of the job ID.

Jobs that are submitted by any user can be specified here without using the `-u` option. If you use the reserved job ID 0, all the jobs that satisfy other options (that is, `-m`, `-q`, `-u`, and `-J` options) are operated on. All other job IDs are ignored.

The options `-u`, `-q`, `-m`, and `-J` have no effect if a job ID other than 0 is specified. Job IDs are returned at job submission time. Use the **bjobs** command to find the job IDs.

Any jobs or job arrays that are killed are logged in the `lsb.acct`file.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Examples

```
bkill -s 17 -q night
```

Sends signal 17 to the last job that was submitted by the invoker to queue `night`.

```
bkill -q short -u all 0
```

Kills all the jobs that are in the queue `short`.

```
bkill -r 1045
```

Forces the removal of unkillable job 1045.

```
bkill -sla Tofino 0
```

Kills all jobs that belong to the service class named `Tofino`.

```
bkill -g /risk_group 0
```

Kills all jobs in the job group `/risk_group`.

```
bkill -app fluent
```

Kills the most recently submitted job that is associated with the application profile `fluent` for the current user.

```
bkill -app fluent 0
```

Kills all jobs that are associated with the application profile `fluent` for the current user.

See also

bsub, **bjobs**, **bqueues**, **bhosts**, **bresume**, **bapp**, **bsla**, **bstop**, **bgadd**, **bgdel**, **bjgroup**, **bparams**, `lsb.params`, `lsb.serviceclasses`, **mbatchd**, **kill**, **signal**

Chapter 22. bladmin

Administrative tool for IBM Spectrum LSF License Scheduler.

Synopsis

`bladmin` *subcommand*

`bladmin` [`-h` | `-V`]

Description

The **bladmin** command provides a set of subcommands to control LSF License Scheduler.

You must be root or a LSF License Scheduler administrator to use this command.

Subcommand synopsis

`ckconfig`

`reconfig` [*host_name* ... | `all`]

`shutdown` [*host_name* ... | `all`]

`blddebug` [`-c` *class_name* ...] [`-l` *debug_level*] [`-f` *logfile_name*] [`-o`]

`blcdebug` [`-l` *debug_level*] [`-f` *logfile_name*] [`-o`] *collector_name* ... | all

`-h`

`-V`

Usage

ckconfig
Checks LSF License Scheduler configuration in the `LSF_ENVDIR/lsf.licensescheduler` and `lsf.conf` files.

By default, the **bladmin ckconfig** command displays only the result of the configuration file check. If warning errors are found, the **bladmin** command prompts you to enter the y option to display detailed messages.

reconfig [*host_name* ... | all]
Reconfigures LSF License Scheduler.

shutdown [*host_name* ... | all]
Shuts down LSF License Scheduler.

blddebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o]
Sets the message log level for the **bld** daemon to include extra information in log files. You must be root or the LSF administrator to use this command.

If the **bladmin blddebug** command is used without any options, the following default values are used:

***class_name*=0**
No additional classes are logged.

***debug_level*=0**
LOG_DEBUG level in the **LS_LOG_MASK** parameter.

logfile_name*=*daemon_name*.log.*host_name
Current LSF system log file in the LSF system log file directory.

-c *class_name* ...
Specifies software classes for which debug messages are to be logged.

Format of *class_name* is the name of a class, or a list of class names separated by spaces and enclosed in quotation marks. Classes are also listed in the `lsf.h` file.

The following log classes are supported:

LC_AUTH
Log authentication messages.

LC_COMM
Log communication messages.

LC_FLEX
Log everything that is related to FLEX_STAT or FLEX_EXEC Flexera APIs.

LC_LICENCE
Log license management messages.

LC_PREEMPT
Log preemption policy messages.

LC_RESREQ
Log resource requirement messages.

LC_TRACE
Log significant program walk steps.

LC_XDR
Log everything that is transferred by XDR.

The default debug level is `0` (no additional classes are logged).

-l *debug_level*
Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

The following values are supported:

0
LOG_DEBUG level in parameter **LS_LOG_MASK** in the `lsf.conf` file. This is the default value.

1
LOG_DEBUG1 level for extended logging.

2
LOG_DEBUG2 level for extended logging.

3
LOG_DEBUG3 level for extended logging.

-f *logfile_name*
Specifies the name of the file where debugging messages are logged. The file name can be a full path. If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file has the following format:

logfile_name.daemon_name.log.*host_name*

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

The default is the current LSF system log file in the LSF system log file directory.

-o
Turns off temporary debug settings and resets them to the daemon starting state. The message log level is reset back to the value of LS_LOG_MASK and classes are reset to the value of LSB_DEBUG_BLD. The log file is also reset to the default log file.

blcdebug [-l *debug_level*] [-f *logfile_name*] [-o] *collector_name* | all

Sets the message log level for the **blcollect** command to include additional information in log files. You must be root or the LSF administrator to use this command.

If the **bladmin blcdebug** command is used without any options, the following default values are used:

***debug_level*=0**

The LOG_DEBUG level in the **LS_LOG_MASK** parameter.

logfile_name*=*daemon_name*.log.*host_name

Current LSF system log file in the LSF system log file directory.

***collector_name*=default**

The default collector name.

-l *debug_level*

Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

The following values are supported:

0

LOG_DEBUG level in parameter **LS_LOG_MASK** in the lsf.conf file. This is the default value.

1

LOG_DEBUG1 level for extended logging.

2

LOG_DEBUG2 level for extended logging.

3

LOG_DEBUG3 level for extended logging.

-f *logfile_name*

Specifies the name of the file where debugging messages are logged. The file name can be a full path. If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file has the following format:

logfile_name.daemon_name.log.*host_name*

On UNIX, if the specified path is not valid, the log file is created in the /tmp directory.

On Windows, if the specified path is not valid, no log file is created.

The default is the current LSF system log file in the LSF system log file directory.

-o

Turns off temporary debug settings and resets them to the daemon starting state. The message log level is reset back to the value of LS_LOG_MASK and classes are reset to the value of LSB_DEBUG_BLD. The log file is also reset to the default log file.

If a collector name is not specified, default value is to restore the original log mask and log file directory for the default collector.

***collector_name* ... | all**

Specifies the collector names, separated by blanks. The all keyword means all the collectors.

-h

Prints command usage to stderr and exits.

-V

Prints release version to stderr and exits.

See also

blhosts, lsf.licensescheduler, lsf.conf

Chapter 23. blaunch

Launches parallel tasks on a set of hosts.

Synopsis

`blaunch` [`-n`] [`-u` *host_file* | `-z` *host_name* ... | *host_name*] [`-use-login-shell` | `-no-shell`] *command* [*argument* ...]

`blaunch` [`-h` | `-V`]

Description

Important: You cannot run **blaunch** directly from the command line.

Restriction: The command **blaunch** does not work with user account mapping. Do not run **blaunch** on a user account mapping host.

Most MPI implementations and many distributed applications use the **rsh** and **ssh** commands as their task launching mechanism. The **blaunch** command provides a drop-in replacement for the **rsh** and **ssh** commands as a transparent method for launching parallel applications within LSF.

The **blaunch** command supports the following core command line options as **rsh** and **ssh**:

- **rsh** *host_name command*
- **ssh** *host_name command*

All other **rsh** and **ssh** command options are silently ignored.

The **blaunch** command transparently connects directly to the RES and **sbatchd** daemons on the remote host. Remote tasks are created and tracked, and the connection back to LSF is maintained. You do not need to insert the **pam** or **taskstarter** commands, or any other wrapper.

The **blaunch** command works only under LSF. It can be used only to launch tasks on remote hosts that are part of a job allocation. It cannot be used as a stand-alone command.

When no host names are specified, LSF runs tasks on all allocated hosts, one remote task per job slot.

On Windows, the **blaunch** command is supported on Windows 2000 or later with the following exceptions:

- Only the following signals are supported: SIGKILL, SIGSTOP, SIGCONT.
- The `-n` option is not supported.
- The `CMD.EXE /C` *<user_command_line>* command is used as intermediate command shell when the `-no-shell` option is not specified.
- The `CMD.EXE /C` command is not used when the `-no-shell` option is specified.
- Windows User Account Control must be configured correctly to run jobs.
- Any tasks killed outside of LSF (for example, with the **taskkill** command) are assumed to have a normal exit.

For parallel jobs launched with the **blaunch** command, job control actions that are defined in the **JOB_CONTROLS** parameter in the `lsb.queues` file take effect only on the first execution host. Job control actions that are defined in the queue do not affect tasks that are running on other hosts. If the **JOB_CONTROLS** parameter is defined, the default job control signals of LSF (SUSPEND, RESUME, TERMINATE) do not reach each task on each execution host.

For parallel jobs launched with the **blaunch** command, automatic job resizing is a signaling mechanism only. It does not expand the extent of the original job that is launched with the **blaunch** command. The resize notification script is required along with a signal-listening script. The signal-listening script runs more **blaunch** commands on notification to allocate the resized resources to make them available to the job tasks. For help creating signal listening and notification scripts, contact IBM Support.

Options

-n
Standard input is taken from `/dev/null`. Not supported on Windows.

-u *host_file*
Runs the task on all hosts that are listed in the *host_file* file.

Specify the path to a file that contains a list of host names. Each host name must be listed on a separate line in the host list file.

This option is exclusive of the `-z` option.

host_name
The name of the host where remote tasks are to be launched.

-z *host_name* ...
Runs the task on all specified hosts.

The host name value for **rsh** and **ssh** is typically a single host name, but you can use the `-z` option to specify a space-delimited list of hosts where tasks are started in parallel.

Specify a list of hosts on which to run the task. If multiple host names are specified, the host names must be enclosed by quotation marks (" or ') and separated by white space.

This option is exclusive of the `-u` option.

-use-login-shell
Launches commands through user's login shell.

Applies only to UNIX and Linux hosts.

-no-shell
Launches commands without any intermediate shell.

***command* [*argument* ...]**
Specify the command to run. The command must be the last argument on the command line.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Diagnostics

Exit status is 0 if all commands are run correctly.

See also

`lsb_getalloc`, `lsb_launch` APIs

Chapter 24. blcollect

License information collection daemon for LSF License Scheduler. The **blcollect** daemon collects license usage information.

Synopsis

`blcollect -c` *collector_name* `-m` *host_name [...]* `-p` *license_scheduler_port* [`-i` *lmstat_interval* | `-D` *lmstat_path*] [`-t` *timeout*]

`blcollect` [`-h` | `-V`]

Description

Periodically collects license usage information from the license manager. It queries the license manager for license usage information and passes the information to the LSF License Scheduler daemon (bld). The **blcollect** daemon improves performance by allowing you to distribute license information queries on multiple hosts.

By default, license information is collected from the license manager on one host. Use the **blcollect** daemon to distribute the license collection on multiple hosts.

For each service domain configuration in the `lsf.licensescheduler` file, specify one name for the **blcollect** daemon to use. You can only specify one collector per service domain, but you can specify one collector to serve multiple service domains. You can choose any collector name you want, but must use that exact name when you run the **blcollect** daemon.

Options

-c
Required. Specify the collector name you set in the `lsf.licensescheduler` file. You must use the collector name (`LIC_COLLECTOR`) you define in the `ServiceDomain` section of the configuration file.

-m
Required. Specifies a space-separated list of hosts to which license information is sent. The hosts do not need to be running LSF License Scheduler or a license manager. Use fully qualified host names.

-p
Required. You must specify the LSF License Scheduler listening port, which is set in the `lsf.licensescheduler` file and has a default value of 9581.

-i *lmstat_interval*
Optional. The frequency in seconds of the calls that LSF License Scheduler makes to the **lmstat** command to collect license usage information from the license manager.

The default interval is 60 seconds.

-D *lmstat_path*
Optional. Location of the FlexNet command **lmstat** or the Reprise License Manager command **rlmstat**, overriding the value defined by the **LMSTAT_PATH** or **RLMSTAT_PATH** parameter in the Parameters section of the `lsf.licensescheduler` file.

-t *timeout*
Optional. Timeout value passed to the **lmstat** command, overwriting the value defined by the **LM_STAT_TIMEOUT** parameter in the Parameters or ServiceDomain section of the `lsf.licensescheduler` file.

This option is ignored if the **LM_TYPE=RLM** parameter is defined in the Parameters section of the `lsf.licensescheduler` file.

-h
Prints command usage to `stderr` and exits.

-V
Prints release version to `stderr` and exits.

See also

`lsf.licensescheduler`

Chapter 25. blcstat

Displays dynamic update information from the **blcollect** daemon for LSF License Scheduler.

Synopsis

`blcstat [-l] [collector_name ...]`

`blcstat [-h | -V]`

Description

Displays the time each license collector daemon (**bcollect**) last sent an update to the **bld** daemon, along with the status of each **blcollect** daemon.

Options

-l
Long format. Displays detailed information for each **blcollect** daemon in a multiline format.

collector_name
Displays information only for the specified **blcollect** daemons.

-h
Prints command usage to `stderr` and exits.

-V
Prints the release version to `stderr` and exits.

Output

COLLECTOR_NAME
The name of the license collector daemon as defined by the `LIC_COLLECTOR=license_collector_name` parameter in the `ServiceDomain` sections of the `lsf.licensescheduler` file. By default, the name is `_default_`.

STATUS
The status of the collector.

ok
The collector is working and all license servers can be reached.

-ok
The collector is working, but not all licenses servers can be reached.

unavail
The collector cannot be reached.

LAST_UPD_TIME
The time the last update was received by the **bld** daemon for this collector.

-l Output

The `-l` option displays a long format listing with the following extra fields:

HOST_NAME
The name of the host that is running this collector.

LICENSE_SERVER
The license server that is configured in the `ServiceDomain` section in the `lsf.licensescheduler` file for this collector.

Multiple lines indicate multiple license servers.

Multiple entries in one line, separated by a vertical bar | indicate configured redundant license servers (sharing a license file).

License server has the following states:

reachable
The license server is running and providing information to **lmstat** or **rlmstat** command.

unreachable
The license server is not running, or some other problem blocks the flow of information to the **lmstat** or **rlmstat** command.

unknown
The **blcollect** daemon is down.

FEATURES
The names of features that are running on license servers for this collector.

LMSTAT_INTERVAL
The interval between updates from this collector as set by the **LM_STAT_INTERVAL** parameter in the Parameters or ServiceDomain section of the `lsf.licensescheduler` file, or by the blcollect daemon at collector startup.

LM_TYPE
The types of license manager systems with which this collector can communicate.

See also

blcollect

Chapter 26. blhosts

Displays the names of all the hosts that are running the LSF License Scheduler daemon (`bld`).

Synopsis

`blhosts [-h | -V]`

Description

The list includes the LSF License Scheduler master host and all the candidate LSF License Scheduler hosts running the **bld** daemon.

Options

-h
Prints command usage to `stderr` and exits.

-V
Prints release version to `stderr` and exits.

Output

Prints the names of all the hosts that are running the LSF License Scheduler bld daemon.

For example, the following sample output shows the LSF License Scheduler master host and two candidate LSF License Scheduler hosts running the bld daemon:

```
bld is running on:
master: host1.domain1.com
slave: host2.domain1 host3.domain1
```

See also

blinfo, **blstat**, **bladmin**

Chapter 27. blimits

Displays information about resource allocation limits of running jobs.

Synopsis

`blimits` [`-a`] [`-c`] [`-w`] [`-A` *application_name* ...] [`-fwd` [`-C` *cluster_name* ...]] [`-Lp` "*ls_license_project*"] [`-m` *host_name* | `-m` *host_group*] [`-n` *limit_name* ...] [`-P` *project_name* ...] [`-q` *queue_name* ...] [`-u` *user_name* | `-u` *user_group* ...]

`blimits -h | -V`

Description

Displays current usage of resource allocation limits configured in Limit sections in the `lsb.resources` file:

- Configured limit policy name
- Users (`-u` option)
- Queues (`-q` option)
- Hosts (`-m` option)
- Project names (`-P` option)
- Application profiles (`-A` option)
- Limits (`SLOTS`, `MEM`, `TMP`, `SWP`, `JOBS`, `INELIGIBLE`)
- Limit configuration (`-c` option). This option is the same as the **bresources** command with no options.

Resources that have no configured limits or no limit usage are indicated by a dash (-). Limits are displayed in a `USED/LIMIT` format. For example, if a limit of 10 slots is configured and 3 slots are in use, then the **blimits** command displays the limit for SLOTS as `3/10`.

Note: If no jobs are running against resource allocation limits, LSF indicates that no information can be displayed:

```
No resource usage found.
```

If limits `MEM`, `SWP`, or `TMP` are configured as percentages, both the limit and the amount that is used are displayed in MB. For example, the **lshosts** command displays `maxmem` of 249 MB, and `MEM` is limited to 10% of available memory. If 10 MB out of 25 MB are used, the **blimits** command displays the limit for `MEM` as `10/25` (10 MB `USED` from a 25 MB `LIMIT`).

Limits are displayed for both the vertical tabular format and the horizontal format for `Limit` sections. If a vertical format `Limit` section has no name, the **blimits** command displays `NONAME`*nnn* under the `NAME` column for these limits. The unnamed limits are numbered in the order that the vertical-format `Limit` sections appear in the `lsb.resources` file.

If a resource consumer is configured as `all`, the limit usage for that consumer is indicated by a dash (`-`)

`PER_HOST` slot limits are not displayed. The **bhosts** command displays these limits as `MAX`.

When LSF adds more resources to a running resizable job, the **blimits** command displays the added resources. When LSF removes resources from a running resizable job, the **blimits** command displays the updated resources.

In LSF multicluster capability, the **blimits** command returns the information about all limits in the local cluster.

Limit names and policies are set up by the LSF administrator in the `lsb.resources` file.

Options

-a
Displays information about all resource allocation limits, even if they are not being applied to running jobs. For limits with running jobs, **blimits** displays limit usage information. For limits that do not have any running jobs, **blimits** displays resource limit configuration information.

-c
Displays all resource configurations in the `lsb.resources` file. This option is the same as the **bresources** command with no options.

-fwd [-C *cluster_name* ...]
Displays forward slot allocation limits. Use the `-fwd` option with the `-c` option to display forward slot limit configuration.

In LSF Advanced Edition, the `-fwd -C` *`cluster_name`* option displays forward slot allocation limits for one or more specified clusters. Use the `-fwd -C` option with the `-c` option to display forward slot limit configuration for the specified cluster.

-w
Displays resource allocation limits information in a wide format. Fields are displayed without truncation.

-A *application_name ...*
Displays resource allocation limits for the specified application profiles.

If a list of applications is specified, application names must be separated by spaces and enclosed in quotation marks (") or (').

-Lp "*ls_project_name*"
Displays resource allocation limits for jobs that belong to the specified LSF License Scheduler project.

-m *"host_name|host_group"*
Displays resource allocation limits for the specified hosts. Use quotation marks when you specify multiple hosts names or groups. Without quotation marks, only the first host name or group that is specified is used.

To see the available hosts, use the **bhosts** command.

The following limit information is displayed for host groups:

- If the limits are configured with the HOSTS parameter, the name of the host group is displayed.
- If the limits are configured with the PER_HOST parameter, the names of the hosts that belong to the group are displayed instead of the name of the host group.

Tip: PER_HOST slot limits are not displayed. The **bhosts** command displays these limits as MXJ.

To see a list of host groups, use the **bmgroup** comand.

-n *limit_name ...*
Displays resource allocation limits the specified named `Limit` sections. If a list of limit sections is specified, `Limit` section names must be separated by spaces and enclosed in quotation marks (" or ').

-P *project_name ...*
Displays resource allocation limits for the specified projects.

If a list of projects is specified, project names must be separated by spaces and enclosed in quotation marks (") or (').

-q *queue_name ...*
Displays resource allocation limits for the specified queues.

The command **bqueues** returns a list of queues that are configured in the system, and information about the configurations of these queues.

In LSF multicluster capability, you cannot specify remote queues.

-u *user_name* | -u *user_group* ...
Displays resource allocation limits for the specified users.

If a list of users is specified, user names must be separated by spaces and enclosed in quotation marks (") or ('). You can specify both user names and user IDs in the list of users.

If a user group is specified, displays the resource allocation limits that include that group in their configuration. To see a list of user groups, use the **bugroup** command).

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Output

Configured limits and resource usage for built-in resources (`slots`, `mem`, `tmp`, and `swp` load indices, and running and suspended job limits) are displayed as `INTERNAL RESOURCE LIMITS` separately from custom external resources, which are shown as `EXTERNAL RESOURCE LIMITS`.

Output for resource consumers

The **blimits** command displays the following fields for resource consumers:

NAME
The name of the limit policy as specified by the `Limit` section **NAME** parameter.

USERS
List of user names or user groups on which the displayed limits are enforced, as specified by the `Limit` section parameters **USERS** or **PER_USER**.

User group names have a slash (/) added at the end of the group name.

QUEUES
The name of the queue to which the limits apply, as specified by the `Limit` section parameters **QUEUES** or **PER_QUEUES**.

If the queue was removed from the configuration, the queue name is displayed as `lost_and_found`. Use the **bhist** command to get the original queue name. Jobs in the `lost_and_found` queue remain pending until they are switched with the **bswitch** command into another queue.

In an LSF multicluster capability resource leasing environment, jobs that are scheduled by the consumer cluster display the remote queue name in the format *queue_name*@*cluster_name*. By default, this field truncates at 10 characters, so you might not see the cluster name unless you use the `-w` or `-l` option.

HOSTS
List of hosts and host groups on which the displayed limits are enforced, as specified by the `Limit` section parameters **HOSTS** or **PER_HOSTS**.

Host group names have a slash (/) added at the end of the group name.

Tip: PER_HOST slot limits are not displayed. The **bhosts** command displays these limits as `MXJ`.

PROJECTS
List of project names on which limits are enforced, as specified by the `Limit` section parameters **PROJECTS** or **PER_PROJECT**.

APPS
List of application names on which limits are enforced, as specified by the `Limit` section parameters **APPS** or **PER_APP**.

Output for resource limits

The **blimits** command displays resource allocation limits for the following resources:

SLOTS
Number of slots that are currently used and maximum number of slots that are configured for the limit policy, as specified by the `Limit` section **SLOTS** parameter.

MEM
Amount of memory that is currently used and maximum that is configured for the limit policy, as specified by the `Limit` section **MEM** parameter.

TMP
Amount of tmp space that is currently used and maximum amount of tmp space that is configured for the limit policy, as specified by the `Limit` section **TMP** parameter.

SWP
Amount of swap space that is currently used and maximum amount of swap space that is configured for the limit policy, as specified by the `Limit` section **SWP** parameter.

JOBS
Number of currently running and suspended jobs and the maximum number of jobs that are configured for the limit policy, as specified by the `Limit` section **JOBS** parameter.

INELIGIBLE
Displays the value that has been configured , as specified by the `Limit` section parameter **INELIGIBLE**.

See also

bclusters, **bhosts**, **bhist**, **bmgroup**, **bqueues**, **bugroup**, `lsb.resources`

Chapter 28. blinfo

Displays static LSF License Scheduler configuration information

Synopsis

`blinfo -Lp | -p | -D | -G | -P`

`blinfo [-a [-t` *token_name* `| "`*token_name* `..."]] [-o alpha | total] [-g "`*feature_group* `..."]`

`blinfo -A [-t` *token_name* `| "`*token_name* `..."] [-o alpha | total ] [-g "`*feature_group* `..."]`

`blinfo -C [-t` *token_name* `| "`*token_name* `..."] [-o alpha | total] [-g "`*feature_group* `..."]`

`blinfo [-t` *token_name* `| "`*token_name* `..."] [-o alpha | total] [-g "`*feature_group* `..."]`

`blinfo [ -h | -V]`

Description

By default, displays information about the distribution of licenses that are managed by LSF License Scheduler.

Options for cluster mode and project mode

-a
Shows all information, including information about non-shared licenses (defined by the **NON_SHARED_DISTRIBUTION** parameter) and workload distribution (defined by the **WORKLOAD_DISTRIBUTION** parameter).

You can optionally provide license token names.

The **blinfo -a** command does not display non-shared information for hierarchical project group scheduling policies. Use the **blinfo -G** command to see hierarchical license project group configuration.

-C
Shows the cluster locality information for license features.

You can optionally provide license token names.

-D
Lists the LSF License Scheduler service domains and the corresponding license server hosts.

-g *feature_group* ...
When the **FEATURE_GROUP** parameter is configured for a group of license features in the `lsf.licensescheduler` file, shows only information about the features that are configured in the **FEATURE_LIST** parameter of the specified feature groups. You can specify more than one feature group at one time.

When you specify feature names with the `-t` option, features in the feature list that is defined by the `-t` option and feature groups are both displayed.

Feature groups that are listed with `-g` but not defined in the `lsf.licensescheduler` file are ignored.

-o alpha | total
Sorts license feature information by total tokens.

alpha
Features are listed in descending alphabetical order.

total
Features are sorted by the descending order of the sum of licenses that are allocated to LSF workload from all the service domains that are configured to supply licenses to the feature. Licenses that are borrowed by non-LSF workload are included in this amount.

-p
Displays values of `lsf.licensescheduler` configuration parameters and `lsf.conf` parameters that are related to LSF License Scheduler. Use this option for troubleshooting.

-t *token_name* | "*token_name* ..."
Shows only information about specified license tokens. Use spaces to separate multiple names, and enclose them in quotation marks.

-P
When the LS_FEATURE_PERCENTAGE=Y or LS_ACTIVE_PERCENTAGE=Y parameter is specified, lists the license ownership (if applicable) in percentage.

-h
Prints command usage to `stderr` and exits.

-V
Prints the LSF License Scheduler release version to `stderr` and exits.

Options for project mode only

-A
When **LOCAL_TO** is configured for a feature in the `lsf.licensescheduler` file, shows the feature allocation by cluster locality.

You can optionally provide license token names.

-G
Lists the hierarchical configuration information.

If the **PRIORITY** parameter is defined in the `ProjectGroup` section of the `lsf.licensescheduler` file, shows the priorities of each project.

-Lp
Lists the active projects that are managed by LSF License Scheduler.

The `-Lp` option displays only projects that are associated with configured features.

If the **PRIORITY** parameter is defined in the `ProjectGroup` section of the `lsf.licensescheduler` file, shows the priorities of each project.

Default output

Displays the following fields:

FEATURE
The license name. This name becomes the license token name.

When the **LOCAL_TO** parameter is configured for a feature in the `lsf.licensescheduler` file, the **blinfo** command shows the cluster locality information for the license features.

MODE
The mode of the license.

Cluster
Cluster mode.

Project
Project mode.

SERVICE_DOMAIN
The name of the service domain that provided the license.

TOTAL
The total number of licenses that are managed by FlexNet. This number comes from FlexNet.

DISTRIBUTION
The distribution of the licenses among license projects in the format [*project_name*, *percentage*[/*number_licenses_owned*]]. Distribution shows how many licenses a project is entitled to use when there is competition for licenses. The percentage is calculated from the share that is specified in the configuration file.

All output with the -a option

As default output, plus all other feature-level parameters defined for each feature.

Cluster locality output with the -C option

NAME
The license feature token name.

When the **LOCAL_TO** parameter is configured for a feature in the `lsf.licensescheduler` file, the **blinfo** command shows the cluster locality information for the license features.

LM_LICENSE_NAME
The actual FlexNet feature name, which is the name that is used by FlexNet to identify the type of license. Might be different from the LSF License Scheduler token name if a different **LM_LICENSE_NAME** is specified in the `lsf.licensescheduler` file.

CLUSTER_NAME
The name of the cluster the feature is assigned to.

FEATURE
The license feature name. This name becomes the license token name.

When the **LOCAL_TO** parameter is configured for a feature in the `lsf.licensescheduler` file, the **blinfo** command shows the cluster locality information for the license features.

SERVICE_DOMAIN
The service domain name.

Service Domain Output with the -D option

SERVICE_DOMAIN
The service domain name.

LIC_SERVERS
Names of license server hosts that belong to the service domain. Each host name is enclosed in parentheses, as shown:

(*port_number*@*host_name*)

Redundant hosts, which share the same license manager license file, are grouped as shown:

(*port_number*@*host_name port_number*@*host_name port_number*@*host_name*)

LM
The license manager system that the license servers in the service domain are using.

Parameters Output with the -p option

Displays values set in the Parameters section of the `lsf.licensescheduler` file.

Displays the following parameter values from the `lsf.conf` file:

LS_LOG_MASK or LOG_MASK
Specifies the logging level of error messages for LSF License Scheduler daemons. If the **LS_LOG_MASK** parameter is not defined in the `lsf.licensescheduler` file, the value of the

LSF_LOG_MASK parameter in the `lsf.conf` file is used. If the **LS_LOG_MASK** parameter and the **LSF_LOG_MASK** parameter are not defined, the default is `LOG_WARNING`.

```
LS_LOG_MASK=LOG_DEBUG
```

The following log levels in order from highest to lowest are supported:

- `LOG_WARNING`
- `LOG_DEBUG`
- `LOG_DEBUG1`
- `LOG_DEBUG2`
- `LOG_DEBUG3`

The most important LSF License Scheduler log messages are at the `LOG_WARNING` level. Messages at the `LOG_DEBUG` level are useful only for debugging.

LSF_LIC_SCHED_HOSTS
List of hosts that are candidate LSF License Scheduler hosts. Defined in the `lsf.conf` file.

LSF_LIC_SCHED_PREEMPT_REQUEUE
Specifies whether to requeue or suspend a job whose license is preempted by LSF License Scheduler. Defined in the `lsf.conf` file.

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE
Specifies whether to release the resources of a job that is suspended when its license is preempted by LSF License Scheduler. Defined in the `lsf.conf` file.

LSF_LIC_SCHED_PREEMPT_STOP
Specifies whether to use job controls to stop a job that is preempted. Defined in the `lsf.conf` file.

Allocation output with -A option, project mode

FEATURE
The license name. This name becomes the license token name.

When the **LOCAL_TO** parameter is configured for a feature in the `lsf.licensescheduler` file, the **blinfo** command shows the cluster locality information for the license features.

PROJECT
The LSF License Scheduler project name.

ALLOCATION
The percentage of shares that are assigned to each cluster for a feature and a project.

Hierarchical Output with the -G option, project mode

The following fields describe the values of their corresponding configuration fields in the `ProjectGroup` Section of the `lsf.licensescheduler` file.

GROUP
The project names in the hierarchical grouping and its relationships. Each entry specifies the name of the hierarchical group and its members. The entry is enclosed in parentheses as shown:

(*group* (*member* ...))

SHARES
The shares that are assigned to the hierarchical group member projects.

OWNERSHIP
The number of licenses that each project owns.

LIMITS
The maximum number of licenses that the hierarchical group member project can use at any one time.

NON_SHARED
The number of licenses that the hierarchical group member projects use exclusively.

PRIORITY
The priority of the project if it is different from the default behavior. A larger number indicates a higher priority.

DESCRIPTION
The description of the project group.

Project Output with the -LP option, project mode

List of active LSF License Scheduler projects.

`-Lp` displays only projects that are associated with configured features.

PROJECT
The project name.

PRIORITY
The priority of the project if it is different from the default behavior. A larger number indicates a higher priority.

DESCRIPTION
The description of the project.

Examples

The **blinfo -a** command (project mode) displays both `NON_SHARED_DISTRIBUTION` and `WORKLOAD_DISTRIBUTION` information when they are defined:

```
blinfo -a
FEATURE      SERVICE_DOMAIN  TOTAL  DISTRIBUTION
g1           LS              3      [p1, 50.0%] [p2, 50.0% / 2]
                                    NON_SHARED_DISTRIBUTION
                                    [p2, 2]
                                    WORKLOAD_DISTRIBUTION
                                    [LSF 66.7%, NON_LSF 33.3%]
```

Files

Reads the `lsf.licensescheduler` file

See also

blstat, **blusers**, `lsf.licensescheduler`, `lsf.conf`

Chapter 29. blkill

Terminates an interactive (**taskman**) LSF License Scheduler task.

Synopsis

`blkill` [`-t` *seconds*] *task_ID*

`blkill` [`-h` | `-V`]

Description

Terminates a running or waiting interactive task in LSF License Scheduler

Users can kill their own tasks. You must be a LSF License Scheduler administrator to terminate another user's task.

By default, the **blkill** command notifies the user and waits 60 seconds before it kills the task.

Options

task_ID
Task ID of the task you want to kill.

-t *seconds*
Specify how many seconds to delay before the **blkill** command kills the task. A value of `0` means to kill the task immediately (do not give the user any time to save work).

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF License Scheduler release version to `stderr` and exits.

Chapter 30. blparams

Displays information about configurable LSF License Scheduler parameters that are defined in the files `lsf.licensescheduler` and `lsf.conf`

Synopsis

`blparams` [`-h` | `-V`]

Description

Displays values set in the Parameters section of the `lsf.licensescheduler` file.

Displays the following parameter values from the `lsf.conf` file:

LS_LOG_MASK or LOG_MASK
Specifies the logging level of error messages for LSF License Scheduler daemons. If the **LS_LOG_MASK** parameter is not defined in the `lsf.licensescheduler` file, the value of the **LSF_LOG_MASK** parameter in the `lsf.conf` file is used. If **LS_LOG_MASK** and **LSF_LOG_MASK** are not defined, the default is `LOG_WARNING`.

```
LS_LOG_MASK=LOG_DEBUG
```

The following log levels are supported in order from highest to lowest:

- `LOG_WARNING`
- `LOG_DEBUG`
- `LOG_DEBUG1`
- `LOG_DEBUG2`
- `LOG_DEBUG3`

The most important LSF License Scheduler log messages are at the `LOG_WARNING` level. Messages at the `LOG_DEBUG` level are useful only for debugging.

LSF_LIC_SCHED_HOSTS
List of hosts that are candidate LSF License Scheduler hosts. Defined in the `lsf.conf` file.

LSF_LIC_SCHED_PREEMPT_REQUEUE
Specifies whether to requeue or suspend a job whose license is preempted by LSF License Scheduler. Defined in the `lsf.conf` file.

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE
Specifies whether to release the slot of a job that is suspended when its license is preempted by LSF License Scheduler. Defined in the `lsf.conf` file.

LSF_LIC_SCHED_PREEMPT_STOP
Specifies whether to use job controls to stop a job that is preempted. Defined in the `lsf.conf` file.

Options

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

`lsf.licensescheduler`, `lsf.conf`

Chapter 31. blstat

Displays dynamic license information.

Synopsis

`blstat [-s] [-S] [-D` *service_domain_name* `|` *"service_domain_name ..."*`] [-P][-t` *token_name* `|` *"token_name ..."*`] [-o alpha | total | avail] [-g` *"feature_group ..."*`] [-slots]`

`blstat [-a] [-c` *token_name*`] [-G] [-Lp` *ls_project_name* `|` *"ls_project_name ..."*`]`

`blstat [-D` *service_domain_name* `|` *"service_domain_name ..."*`] [-t` *token_name* `|` *"token_name ..."*`] [-f` *"field_name*`[:[-][`*output_width*`]] ... [delimiter='`*character*`']" ] [-a] [-alloc] [-cname] [-noheader] [-x] [-X] [-R` *"res_req"*`] [`*host_name ...* `|` *host_group ...* `|` *compute_unit ...*`]`

`blstat [-Lp` *ls_project_name* `|` *"ls_project_name ..."*`] [-f` *"field_name*`[:[-][`*output_width*`]] ... [delimiter='`*character*`']" ] [-a] [-alloc] [-cname] [-noheader] [-x] [-X] [-R` *"res_req"*`] [`*host_name ...* `|` *host_group ...* `|` *compute_unit ...*`]`

`blstat [-h | -V]`

Description

Displays license usage statistics for LSF License Scheduler.

By default, shows information about all licenses and all clusters.

Options for cluster mode and project mode

-S
Displays information on the license servers that are associated with license features.

-s
Displays license usage of the LSF and non-LSF workloads. Workload distributions are defined by the **WORKLOAD_DISTRIBUTION** parameter in the `lsf.licensescheduler` file. The **blstat** command marks any distribution policy violations with an asterisk (*) at the beginning of the line.

-D *service_domain_name* | "*service_domain_name ...*"
Shows information only about specified service domains. Use spaces to separate multiple names, and enclose names in quotation marks.

-f *output_format_string*
Sets the customized output format.

- Specify which **blstat** fields (or aliases instead of the full field names), in which order, and with what width to display.
- Specify only the **blstat** field name or alias to set its output to unlimited width and left justification.
- Specify the colon (`:`) without a width to set the output width to the recommended width for that field.
- Specify the colon (`:`) with a width to set the maximum number of characters to display for the field. When the value exceeds this width, **blstat** truncates the output.
- Specify a hyphen (`-`) to set right justification when **blstat** displays the output for the specific field. If not specified, the default is to set left justification.
- Use `delimiter=` to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the **blstat** command with no options, and to output for **blstat** with the following options: `-Lp`, `-D`, `-t`. The `-f` option does not work with any other **blstat** option.

The following are the field names used to specify the **blstat** fields to display, recommended width, aliases you can use instead of field names, the applicable LSF License Scheduler modes, and a brief description of the field:

Table 4. Output fields for blstat

Field name	Width	Alias	Mode	Description
service_domain_name	19	sd_name	all cluster mode	Name of the service domain
service_domain_others	9	sd_others		Total number of "others" tokens in the service domain
service_domain_lsf_total	12	sd_lsf_total		Total number of tokens that LSF can use from the service domain
service_domain_alloc	8	sd_alloc		Total number of allocated tokens from the service domain
service_domain_use	7	sd_use		Total number of used tokens from the service domain

Table 4. Output fields for blstat (continued)

Field name	Width	Alias	Mode	Description
service_domain_inuse	8	sd_inuse		Total number of tokens in use from the service domain
service_domain_reserve	7	sd_rsv		Total number of reserved tokens from the service domain
service_domain_free	7	sd_free		Total number of free tokens from the service domain
project_name	15	proj_name		Name of the project
project_share	7	proj_share		Number of shares for the project
project_own	7	proj_own		Number of tokens that are owned by the project
project_inuse	7	proj_inuse		Number of tokens that are in use by the project
project_reserve	7	proj_rsv		Number of tokens that are reserved by the project
project_free	7	proj_free		Number of free tokens for the project
project_demand	8	proj_demand		Number of tokens that are demanded by the project
project_limit	7	proj_limit		The project limit
project_non_share	11	proj_non_share		Number of non-shared tokens for the project
cluster_name	15	cl_name	all	Name of the cluster
cluster_share	7	cl_share	cluster	Number of shares in the cluster

Modes: project, default, distribution, cluster

Table 4. Output fields for blstat (continued)

Field name	Width	Alias	Mode	Description
cluster_alloc	7	cl_alloc	cluster and fast dispatch mode all	Number of tokens that are allocated to the cluster
cluster_target	8	cl_target		Target number of tokens for the cluster
cluster_inuse	7	cl_inuse		Number of tokens that are in use in the cluster
cluster_reserve	7	cl_rsv		Number of tokens that are reserved in the cluster
cluster_free	7	cl_free		Number of free tokens in the cluster

Table 4. Output fields for blstat (continued)

Field name	Width	Alias	Mode	Description
cluster_over	7	cl_over	clusterandfastdispatchmode	Number of overused tokens in the cluster
cluster_demand	8	cl_demand	all	Demand of the cluster
cluster_peak	7	cl_peak	clusterandfastdispatchmode	Peak number of used tokens in the cluster
cluster_buffer	8	cl_buff		Allocate buffer in the cluster

Table 4. Output fields for blstat (continued)

Field name	Width	Alias	Mode	Description
cluster_acum_use	10	cl_acum_use	project mode	Accumulated number of used tokens in the cluster
cluster_scaled_acum	13	cl_scaled_acum	project mode	Scaled number of accumulated used tokens in the cluster
cluster_avail	7	cl_avail	project mode	Number of available tokens in the cluster
feature_name	18	feat_name	all	Name of the feature
feature_mode	20	feat_mode	all	Feature mode
feature_lm_license_name	17	feat_lm_lic_name	all	Name of the license feature in the FlexNet server

Note: Field names and aliases are not case-sensitive. Valid values for the output width are any positive integer 1 - 4096.

-g *feature_group* ...
When **FEATURE_GROUP** is configured for a group of license features in the `lsf.licensescheduler` file, shows information about features that are configured in the **FEATURE_LIST** parameter for specified feature groups. You can specify more than one feature group.

When you specify feature names with the `-t` option, features in the **FEATURE_LIST** value that is defined by the `-t` option and feature groups are both displayed.

Feature groups that are listed but not defined in the `lsf.licensescheduler` file are ignored.

-slots
Displays how many slots are currently used by LSF License Scheduler jobs (`Current job slots in use`) and the peak number of slots in use (`Peak job slots used`.

-o alpha | total | avail
Sorts license feature information alphabetically by total licenses, or by available licenses.

`alpha`
Features are listed in descending alphabetical order.

`total`
Features are sorted by the descending order of the sum of licenses that are allocated to LSF workload from all the service domains that are configured to supply licenses to the feature. Licenses that are borrowed by non-LSF workload are not included in this amount.

`avail`
Features are sorted by descending order of licenses available, including free tokens.

-P

Displays percentage values for INUSE and RESERVE. The percentage value represents the number of used and reserved tokens that the project has, compared to total number of licenses.

-t *token_name* | "*token_name* ..."
Shows only information about specified license tokens. Use spaces to separate multiple names, and enclose the names in quotation marks.

-h
Prints command usage to `stderr` and exits.

-V
Prints the release version to `stderr` and exits.

Options for project mode only

-a
Displays each project group's accumulated value of licenses. The license token dispatch order is based on the sort order, which is based on the scaled accumulate value of each project. The lower the value, the sooner the license token is dispatched to that project.

-c *token_name*
Displays cross cluster information for tokens.

In project mode, the information is sorted by the value of the **SCALED_ACUM** parameter. The first cluster that is listed receives tokens first.

Information that is displayed includes token usage, reserved tokens, free tokens, demand for tokens, accumulated value of tokens, and scaled accumulate value of tokens in each cluster.

For fast dispatch project mode, also displays the actual and ideal number of tokens that are allocated to the cluster:

TARGET
The ideal number of licenses that are allocated to the cluster.

OVER
The number of licenses that are checked out by RUN jobs in the cluster under the license projects in excess of the `rusage` value.

FREE
The number of license that is allocated to the cluster but not used.

DEMAND
The number of tokens that are required by the cluster under the license project.

-G
Displays dynamic hierarchical license information.

The **blstat -G** command also works with the `-t` option to display only hierarchical information for the specified feature names.

-Lp *ls_project_name* | "*ls_project_name* ..."
Shows project description for specified non-hierarchical projects. Use spaces to separate multiple names and enclose names in quotation marks.

If project group paths are enabled (**PROJECT_GROUP_PATH=Y** parameter in the `lsf.licensescheduler` file), the **blstat -Lp** command displays the license projects that are associated with the specified project for all features. The **blstat -Lp -t** command displays the associated license projects for the specified feature. If the parameter is disabled, only the specified project is displayed.

Output

Information is organized first by license feature, then by service domain. For each combination of license and service domain, LSF License Scheduler displays a line of summary information followed by rows of license project or cluster information.

In each group of statistics, numbers and percentages refer only to licenses of the specified license feature that can be checked out from FlexNet license server hosts in the specified service domain.

Cluster mode summary output

FEATURE
The license name. This name appears only once for each feature.

SERVICE_DOMAIN
The name of the service domain that provided the license.

TOTAL_TOKENS

The number of licenses from this service domain that is reserved for LSF License Scheduler jobs.

TOTAL_ALLOC
The number of licenses from this service domain that are allocated to clusters by LSF License Scheduler.

In most cases **TOTAL_ALLOC** is equal to **TOTAL_USE**, however, when there are licenses counted under **OTHERS** or when tokens are reclaimed, **TOTAL_ALLOC** might be less than **TOTAL_TOKENS**.

TOTAL_USE

The number of licenses in use by LSF License Scheduler projects, which are determined by totaling all **INUSE**, **RESERVE**, and **OVER** values.

OTHERS
The number of licenses that are checked out by applications outside of LSF License Scheduler.

Cluster output for cluster mode

For each cluster that is configured to use the license, the `blstat` command displays the following information.

CLUSTER
The cluster name.

SHARE
The percentage of licenses that are assigned to the license project by the LSF License Scheduler administrator. The share determines how many licenses the project is entitled to when projects compete for licenses. This information is static, and for a LAN service domain is always 100%.

The percentage is calculated to one decimal place by using the share assignment in the `lsf.licensescheduler` file.

ALLOC
The number of licenses that are currently allocated to the cluster by the bld daemon.

TARGET
The ideal number of licenses that are allocated to the cluster. Normally, this amount is the same as the ALLOC field, but the values might temporarily be different. For example, when a license is reclaimed, where one cluster is using more than its allocation, another cluster is prevented from getting its ideal amount.

INUSE
The number of licenses that are checked out by jobs in the cluster.

RESERVE
The number of licenses that are reserved in the service domain for jobs that are running in the cluster. This value is the difference between the job `rusage` and the number of checked out licenses attributed to the job by LSF License Scheduler.

If the same license is available from both LAN and WAN service domains in cluster mode, LSF License Scheduler expects jobs to try to obtain the license from the LAN first. The administrator must make sure that applications behave in this manner, by using the FlexNet environment variables and **LM_LICENSE_FILE**.

OVER
The amount of license check-outs exceeding `rusage`, summed over all jobs.

PEAK
By default, the maximum of **INUSE**+**RESERVE**+**OVER** observed over the past 5 minutes. The observation period is set by the **PEAK_INUSE_PERIOD** parameter in either the **Parameters** or **Feature** section.

PEAK is used in scheduling to estimate the cluster's capacity to use licenses in this service domain.

BUFFER
The optional allocation buffer that is configured in the **ALLOC_BUFFER** parameter of the **Feature** section for WAN service domains. When defined, dynamic license token allocation is enabled.

FREE
The number of licenses the cluster has free. The license tokens are allocated to the license project by LSF License Scheduler, but the licenses are not reserved and are not yet checked out from the FlexNet license manager.

DEMAND
Numeric value that indicates the number of tokens that are required by each cluster.

Project mode summary output

FEATURE
The license name. This name appears only once for each feature.

SERVICE_DOMAIN
The name of the service domain that provided the license.

TOTAL_INUSE
The number of licenses in use by LSF License Scheduler projects. Represents licenses in use that are checked out from the FlexNet license manager.

TOTAL_RESERVE
The number of licenses that are reserved for LSF License Scheduler projects. Represents licenses that are reserved and are not yet checked out from the FlexNet license manager.

TOTAL_FREE
The number of free licenses that are available to LSF License Scheduler projects. Represents licenses that are not reserved or in use.

OTHERS

The number of licenses that are checked out by users who are not submitting their jobs to LSF License Scheduler projects.

By default, in project mode these licenses are not being managed by LSF License Scheduler policies.

(Project mode only) To enforce license distribution policies for these license features, configure the ENABLE_DYNAMIC_RUSAGE=Y parameter in the **Feature** section for those features in the `lsf.licensescheduler` file.

Workload output for both modes

LSF_USE
The total number of licenses in use by LSF License Scheduler projects in the LSF workload.

LSF_DESERVE
The total number of licenses that are assigned to LSF License Scheduler projects in the LSF workload.

LSF_FREE
The total number of free licenses available to LSF License Scheduler projects in the LSF workload.

NON_LSF_USE
The total number of licenses in use by projects in the non-LSF workload.

NON_LSF_DESERVE
The total number of licenses that are assigned to projects in the non-LSF workload.

NON_LSF_FREE
The total number of free licenses available to projects in the non-LSF workload.

Project output for project mode

For each project that is configured to use the license, the `blstat` command displays the following information.

PROJECT
The LSF License Scheduler project name.

SHARE
The percentage of licenses that are assigned to the license project by the LSF License Scheduler administrator. The share value determines how many licenses the project is entitled to when projects compete for licenses. This information is static.

The percentage is calculated to one decimal place by using the share assignment in the `lsf.licensescheduler` file.

LIMITS
The maximum number of licenses that the hierarchical project group member project can use at any one time.

OWN
Numeric value that indicates the number of tokens that are owned by each project.

INUSE
The number of licenses in use by the license project. Licenses in use are checked out from the FlexNet license manager.

RESERVE
The number of licenses that are reserved for the license project. The corresponding job is running, but its license is not yet checked out from the FlexNet license manager.

FREE
The number of licenses the license project has free. The license tokens are allocated to the license project by LSF License Scheduler, but the licenses are not reserved and are not yet checked out from the FlexNet license manager.

DEMAND
Numeric value that indicates the number of tokens that are required by each project.

NON_SHARED
The number of non-shared licenses that belong to the license project. The license tokens that are allocated to non-shared distribution are scheduled before the tokens allocated to shared distribution.

DESCRIPTION
Description of the project.

ACUM_USE
The number of tokens that are accumulated by each consumer at run time. It is the number of licenses that are assigned to a consumer for a specific feature.

SCALED_ACUM
The number of tokens that are accumulated by each consumer at run time, divided by the SHARE value. LSF License Scheduler uses this value to schedule the tokens for each project.

Cross cluster token output for project mode

For each project that is configured to use the license, the `blstat -c` command displays the following information.

PROJECT
The LSF License Scheduler project name.

CLUSTER
The name of a cluster that uses the project.

INUSE
The number of licenses in use by the license project. Licenses in use are checked out from the FlexNet license manager.

RESERVE
The number of licenses that are reserved for the license project. The corresponding job is running, but its license is not yet checked out from the FlexNet license manager.

FREE
The number of licenses the license project has free. The license tokens are allocated to the license project by LSF License Scheduler, but the licenses are not reserved and are not yet checked out from the FlexNet license manager.

NEED
The total number of tokens that are required by pending jobs (`rusage`).

ACUM_USE
The number of tokens that are accumulated by each consumer at run time. It is the number of licenses that are assigned to a consumer for a specific feature.

SCALED_ACUM
The number of tokens that are accumulated by each consumer at run time, divided by the SHARE value. LSF License Scheduler uses this value to schedule the tokens for each project.

Cross cluster token output for fast dispatch project mode

For each project in fast dispatch project mode that is configured to use the license, `blstat -c` displays the following information.

PROJECT
The LSF License Scheduler project name.

CLUSTER
The name of a cluster that uses the project.

ALLOC
The actual number of licenses that are currently allocated to the cluster. It is possible that the sum of licenses in the INUSE, RESERVE, and OVER fields are larger than the ALLOC field. In this case, the number of tokens that the cluster occupies will eventually decrease towards the ALLOC value after the job finishes.

The percentage is calculated to one decimal place by using the share assignment in the `lsf.licensescheduler` file.

TARGET
The ideal number of licenses that are allocated to the cluster. Normally, this amount is the same as the ALLOC field, but the values might temporarily be different. For example, a license is reclaimed, where one cluster is using more than its allocation, which prevents another cluster from getting its ideal amount.

INUSE
The number of licenses in use by the cluster under the license project. Licenses in use are checked out from the FlexNet license manager.

RESERVE
The number of licenses that are reserved by jobs in the cluster under the license project. The corresponding job is running, but its license is not yet checked out from the FlexNet license manager. The INUSE and RESERVE fields add up to the `rusage` of RUN jobs in the cluster.

OVER
The number of licenses that are checked out by RUN jobs in the cluster under the license project in excess of the `rusage` value.

FREE
The number of licenses that the cluster under the license project has free. The license tokens are allocated to the license project by LSF License Scheduler, but the licenses are not reserved and are not yet checked out from the FlexNet license manager.

DEMAND
Numeric value reported from the cluster, which indicates the number of tokens that are required by the cluster under the license project.

Project group output for project mode

SHARE_INFO_FOR
The root member and name of the hierarchical project group. The project information that is displayed after this title shows the information specific to this particular project group. If this root member is itself a member of another project group, the relationship is displayed as follows:

/root_name/member_name/...

PROJECT/GROUP
The members of the hierarchical group, which is listed by group or project name.

-slots output

Displays the following information:

Current job slots in use
The total number of slots currently being used by LSF License Scheduler jobs, including **taskman** jobs.

Peak job slots used
The peak number of slots in use since the last time LSF License Scheduler was restarted.

Viewing license feature locality

In project mode, when the **LOCAL_TO** parameter is configured for a feature in the `lsf.licensescheduler` file, the **blstat** command shows the cluster locality information for the license features.

Sample output

For example, the **blstat** command has the following output for a cluster mode feature:

```
blstat -t f1000
FEATURE: f1000
 SERVICE_DOMAIN: Lan12
 TOTAL_TOKENS: 1000 TOTAL_ALLOC: 967  TOTAL_USE: 655  OTHERS: 25
  CLUSTER     SHARE   ALLOC TARGET INUSE RESERVE OVER  PEAK  BUFFER FREE  DEMAND
  clusterA     66.7 %  647  15     0     655     0     658   100    0     7452
  clusterB     33.3 %  320  15     0     0       0     0     -      320   0
 SERVICE_DOMAIN: Lan99
 TOTAL_TOKENS: 2000 TOTAL_ALLOC: 2000 TOTAL_USE: 0    OTHERS: 0
  CLUSTER     SHARE   ALLOC TARGET INUSE RESERVE OVER  PEAK  BUFFER FREE  DEMAND
  clusterA     25.0 %  500  15     0     0       0     0     100    500   0
  clusterB     25.0 %  500  15     0     0       0     0     100    500   0
  clusterC     25.0 %  500  15     0     0       0     0     -      500   0
  clusterD     25.0 %  500  15     0     0       0     0     -      500   0
```

For example, for a project mode feature with a group distribution configuration the **blstat** command shows the locality of the `hspice` feature that is configured for various sites:

```
blstat
FEATURE: hspice
 SERVICE_DOMAIN: SD3 SD4
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 22   OTHERS: 0
  PROJECT                SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                    50.0 %   3     1    0        0     11
  Lp2                    50.0 %   1     3    0        0     11
FEATURE: hspice@clusterA
 SERVICE_DOMAIN: SD1
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 25   OTHERS: 0
  PROJECT                SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                    50.0 %   4     0    0      12       3
  Lp2                    50.0 %   5     0    0      13       1
FEATURE: hspice@siteB
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 65   OTHERS: 0
  PROJECT                SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                    50.0 %   4     0    0      32       2
  Lp2                    50.0 %   5     0    0      33       6
```

For example, for a project mode feature, the **blstat -c** command displays the following information:

```
blstat -c f50
FEATURE: f50
PROJECT         CLUSTER        INUSE   RESERVE   FREE   NEED   ACUM_USE   SCALED_ACUM   AVAIL
myProj2         interactive    0       0         9      0      0.0        0.0           9
                clusterA       0       0         8      0      0.0        0.0           0
                clusterB       0       0         8      0      0.0        0.0           0
default         interactive    0       0         9      0      0.0        0.0           9
                clusterA       0       0         8      0      0.0        0.0           0
                clusterB       0       0         8      0      0.0        0.0           0
```

For example, for a fast dispatch project mode feature, the **blstat -c** command displays the following information:

```
blstat -c f100
FEATURE: f100
PROJECT     CLUSTER         ALLOC TARGET INUSE  RESERVE OVER  FREE  DEMAND
myProj1     interactive     4     4      0      0       0     4     0
            clusterA        3     3      0      0       0     3     0
            clusterB        3     3      0      0       0     3     0
myProj2     interactive     30    30     0      0       0     30    0
            clusterA        30    30     0      0       0     30    0
            clusterB        30    30     0      0       0     30    0
```

See also

blhosts, **blinfo**

Chapter 32. bltasks

Displays LSF License Scheduler interactive task information.

Synopsis

`bltasks [-l] [task_ID]`

`bltasks [-l] [-p | -r | -w] [-Lp "ls_project_name..."] [-m "host_name..."] [-t "terminal_name..."] [-u "user_name..."]`

`bltasks [| -h | -V]`

Description

Displays current information about interactive tasks that were submitted by using **taskman**.

By default, displays information about all tasks that are managed by LSF License Scheduler.

Options

task_ID
Displays information only about the specified task.

-l
Long format. Displays detailed information for each task in a multiline format.

-p
Displays information only about tasks with PREEMPTED status.

Cannot be used with the `-r` or `-w` option.

-r
Displays information only about tasks with RUN status.

Cannot be used with the `-p` or `-w` option.

-w
Displays information only about tasks with `WAIT` status.

Cannot be used with the `-p` or `-r` option.

-Lp "*ls_project_name...*"
Displays information only about tasks that are associated with the specified projects.

If project group paths are enabled (the `PROJECT_GROUP_PATH=Y` parameter in the `lsf.licensescheduler` file) and a task has multiple effective license projects, displays only the first task that is associated with the specified effective license project.

-m "*host_name...*"
Displays information only about tasks that are submitted from the specified hosts.

-t "*terminal_name...*"
Displays information only about tasks that are submitted from the specified terminals.

-u "*user_name...*"
Displays information only about tasks that are submitted by the specified users.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF License Scheduler release version to `stderr` and exits.

Default Output

Displays the short format with the following information:

TID
Task ID that LSF License Scheduler assigned to the task.

USER
The user who submitted the task.

STAT
The status of the task.

RUN
Task is running.

WAIT
Task is not yet started.

PREEMPT
Task was preempted and currently has no license token.

HOST
The name of host from which the task was submitted.

PROJECT
The name of the project to which the task belongs.

FEATURES
Name of the LSF License Scheduler token.

CONNECT TIME

The submission time of the task.

EFFECTIVE_PROJECT
The actual project that the job used. If group project paths are enabled with the PROJECT_GROUP_PATH=Y parameter in the `Parameters` section of the `lsf.licensescheduler` file, LSF License Scheduler attempts to calculate a proper project according to the configuration if the license project does not exist or is not authorized for the features. Otherwise, the submission license project is the effective license project.

Output for -l Option

Displays detailed information for each task in multi-line format. If the task is in `WAIT` status, the **bltasks** command displays `The application manager is waiting for a token to start` and the resource requirement. Otherwise, the following current resource usage of task is displayed:

TERMINAL
The terminal the task is using.

PGID
UNIX process group ID.

CPU
The total accumulated CPU time of all processes in a task, in seconds.

MEM
Total resident memory usage of all processes in a task, in KB.

SWAP
Total virtual memory usage of all processes in a task, in KB.

Keyboard idle since
Time at which the task became idle.

RES_REQ
The resource requirement of the task.

Command line
The command the LSF License Scheduler task manager is running.

Chapter 33. blusers

Displays license usage information for LSF License Scheduler.

Synopsis

`blusers` [`-J` [`-u` *user_name*]] [`-t` *token_name*...] [`-l`]

`blusers -P` [`-lm`] `-j` *job_ID* `-u` *user_name* `-m` *host_name* [`-c` *cluster_name*]

`blusers` [`-h` | `-V`]

Description

By default, displays summarized information about usage of licenses.

Options

-J
Displays detailed license resource request information about each job.

In cluster mode, **`blusers -J`** displays tokens for features where the job checks out licenses in excess of the `rusage` value, which are tokens that are checked out to features that a job did not explicitly request. These features have an `INUSE` value, but no `RUSAGE` value.

-u *user_name*
Displays detailed license resource request information about each job that belongs to the single user specified.

-t
Displays detailed license resource request information about each job that uses the token names specified.

-l
Long format. Displays more license usage information.

-P [-lm] -j *job_ID* -u *user_name* -m *host_name*
-P [-lm] -c *cluster_name* -j *job_ID* -u *user_name* -m *host_name*
This string of options is used in a customized preemption script. To identify a job, specify the LSF job ID, the user name, the name of the host where the job is running, and the cluster name.

If the job is an interactive task that is submitted with the `taskman` command, do not specify the `-c` *`cluster_name`* option.

You see the display terminal that is used by the job, its checked out licenses, and the license servers that provided the licenses. Each license feature from each license server has one line of output, in the following format:

`port_number@host_name token_name user_name host_name display`

Specify the `-lm` option to display the license manager system type and the necessary information that is used in a customized preemption script. The `-lm` option is required to display information on Reprise License Manager features. Each license feature from each license server has one line of output, in the following format:

`FLEXLM` *`port_number@host_name token_name user_name host_name display`*

`RLM` *`host_name port_number vendor handle`*

If you use Reprise License Manager and you do not specify the `-lm` option, the output for the Reprise License Manager feature is `NO TTY` *`feature_name`*.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF License Scheduler release version to `stderr` and exits.

Default output

FEATURE
The license name. This feature name becomes the license token name.

SERVICE_DOMAIN
The name of the service domain that provided the license.

USER
The name of the user who submitted the jobs.

HOST
The name of the host where jobs started.

NLICS
The number of licenses that are checked out from FlexNet.

NTASKS
The number of running tasks that use these licenses.

-J output

The `-J` option displays the following summary information for each job:

JOBID
The job ID assigned by LSF.

USER
The name of the user who submitted the job.

HOST
The name of the host where the job started.

PROJECT
The name of the license project that the job is associated with.

CLUSTER
The name of the LSF cluster that the job is associated with. Displays a dash (`-`) for an interactive job.

START_TIME
The job start time.

The `-J` option displays the following information for each license in use by the job:

RESOURCE
The name of the license that is requested by the job.

RUSAGE
The number of licenses that are requested by the job.

SERVICE_DOMAIN
The name of the service domain that provided the license.

The keyword `UNKNOWN` means that the job requested a license from LSF License Scheduler but did not check out the license from FlexNet.

INUSE
The number of checked out licenses. Displays a dash (`-`) when the `SERVICE_DOMAIN` value is `UNKNOWN`.

EFFECTIVE_PROJECT
The actual project that the job used. If group project paths are enabled (the `PROJECT_GROUP_PATH=Y` parameter in the Parameters section of the `lsf.licensescheduler` file), LSF License Scheduler attempts to calculate a proper project according to the configuration if the license project does not exist or is not authorized for the feature. Otherwise, the submission license project is the effective license project.

Long output with the -l option

The `-l` option displays the default output and the following additional information for each job:

OTHERS
License usage for non-managed or non-LSF workload.

DISPLAYS
Terminal display that is associated with the license feature. When Reprise License Manager is the license manager (the LM_TYPE=RLM parameter in the `lsf.licensescheduler` file), this field displays no information: `()`.

Viewing license feature locality

When the **LOCAL_TO** parameter is configured for a feature in the `lsf.licensescheduler` file, the **blusers** command shows the cluster locality information for the license features:

```
blusers
FEATURE          SERVICE_DOMAIN  USER     HOST       NLICS    NTASKS
hspice@clusterA  SD1             user1    host1      1        1
hspice@siteB     SD2             user2    host2      1        1
```

Examples

```
blusers -l
FEATURE  SERVICE_DOMAIN  USER   HOST    NLICS   NTASKS OTHERS  DISPLAYS
feat1    LanServer       user1  hostA   1       1      0     (/dev/tty)
```

```
blusers -J
JOBID   USER      HOST      PROJECT             CLUSTER        START_TIME
553     user1     hostA     project3            cluster1       Oct  5 15:47:14
RESOURCE        RUSAGE        SERVICE_DOMAIN   INUSE  EFFECTIVE_PROJECT
feature1        1             SD1              1      /group2/project3
feature2        1             SD1              1      /group2/others
feature3        -             SD1              1      /group2/project3
```

```
blusers -P -lm -j 101 -u user1 -m hostA -c cluster1
FLEXLM 19999@server f1 user1 hostA /dev/tty
RLM server1 5053 demo 41
```

See also

blhosts, **blinfo**, **blstat**

Chapter 34. bmgroup

Displays information about host groups and compute units.

Synopsis

`bmgroup [-r] [-l] [-w] [-cu] [-cname] [group_name... | cluster_name]`

`bmgroup [-h | -V]`

Description

Displays compute units, host groups, host names, and administrators for each group or unit. Host group administrators are expanded to show individual user names even if a user group is the configured administrator. Group administrators who are inherited from member subgroups are also shown.

By default, displays information about all compute units, host partitions, and host groups, including host groups created for EGO-enabled SLA scheduling.

Leased-in hosts are displayed as a list of hosts in the form `host_name@cluster_name` (for example, `hosta@cluster1`). If the **LSB_BMGROUP_ALLREMOTE_EXPAND=N** parameter is configured in the `lsf.conf` file or as an environment variable, leased-in hosts are represented by a single keyword `allremote` instead of being displayed as a list.

Host groups for EGO-enabled SLA

When hosts are allocated to an EGO-enabled SLA, they are dynamically added to a host group created by the SLA. The name of the host group is `_sla_`*sla_name*, where *sla_name* is the name of the EGO-enabled SLA defined in the `lsb.serviceclasses` file or in the **ENABLE_DEFAULT_EGO_SLA** parameter in the `lsb.params` file. One of the hosts in the host group has the name `_virtual`.

When the host is released to EGO, the entry is removed from the host group. The **bmgroup** command displays the hosts that are allocated by EGO to the host group created by the SLA.

Options

-l
Displays static and dynamic host group members. A plus sign (+) before the host name indicates that the host is dynamic and is a member of the compute unit, host partitions, or host group. A minus sign (-) before the host name indicates that the host is not an EGO host but is a member of the dynamic group, partition, or unit.

Also identifies condensed compute units, host partitions, or host groups as defined by **CONDENSE** in the `HostGroup` or `ComputeUnit` section of the `lsb.hosts` file.

-r
Expands host groups, host partitions, and compute units recursively. The expanded list contains only host names; it does not contain the names of subgroups. Duplicate names are listed only once.

-w
Wide format. Displays names without truncating fields.

-cname
In LSF Advanced Edition, includes the cluster name for execution cluster host groups in output. The output is sorted by cluster and then by host group name.

-cu
Displays compute unit information. Use with *cu_name*]to display only the specified compute unit.

group_name
Displays only information about the specified host groups (or compute unit with the `-cu` option). Do not use quotation marks when you specify multiple groups.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Output

In the list of hosts, a name followed by a slash (/) indicates a subgroup.

Files

Host groups, compute units, and host partitions are configured in the configuration file `lsb.hosts`. Compute unit types are defined by the **COMPUTE_UNIT_TYPES** parameter in the `lsb.params` file.

See also

`lsb.hosts`, `lsb.params`, **bugroup**, **bhosts**

Chapter 35. bmig

Migrates checkpointable or rerunnable jobs.

Synopsis

`bmig` [`-f`] [*job_ID* | "*job_ID*[*index_list*]"] ...

`bmig` [`-f`] [`-J` *job_name*]

[`-m` "*host_name* `...` " | `-m` "*host_group* `...` "] [`-u` *user_name* | `-u` *user_group* | `-u` `all`] [`0`]

`bmig` [`-h` | `-V`]

Description

Migrates one or more of your checkpointable or rerunnable jobs to a different host. You can migrate only running or suspended jobs. You cannot migrate pending jobs. Members of a chunk job in the `WAIT` state can be migrated. LSF removes waiting jobs from the job chunk and changes their original dispatch sequence.

By default, migrates the most recently submitted job, or the most recently submitted job that also satisfies other specified options (`-u` and `-J`). Specify `0` (zero) to migrate multiple jobs. Only LSF administrators and root can migrate jobs that are submitted by other users. Both the original and the new hosts must have the following characteristics:

- Be binary compatible
- Run the same version of the operating system for predictable results
- Have network connectivity and read/execute permissions to the checkpoint and restart executable files (in the `LSF_SERVERDIR` directory by default)
- Have network connectivity and read/write permissions to the checkpoint directory and the checkpoint file
- Have access to all files open during job execution so that LSF can locate them using an absolute path name

When you migrate a checkpointable job, LSF checkpoints and kills the job and then restarts the job on the next available host. If checkpoint fails, the job continues to run on the original host. If you use the **bmig** command while a job is being check pointed, for example, with periodic checkpointing enabled, LSF ignores the migration request.

When you migrate a rerunnable job, LSF kills the job and restarts it from the beginning on the next available host. LSF sets the environment variable **LSB_RESTART** to Y when a migrating job restarts or reruns.

Note: The job owner does not receive notification when LSF kills a checkpointable or rerunnable job as part of job migration.

In LSF multicluster capability, you must use the **brun** command rather than the **bmig** command to move a job to another host.

When absolute job priority scheduling (APS) is configured in the queue, LSF always schedules migrated jobs before pending jobs. For migrated jobs, LSF keeps the existing job priority. If the **LSB_REQUEUE_TO_BOTTOM** and **LSB_MIG2PEND** parameters are configured in the `lsf.conf` file, the migrated jobs keep their APS information. The migrated jobs compete with other pending jobs based on the APS value. If you want to reset the APS value, you must use the **brequeue** command instead of the **bmig** command.

Options

-f
Forces a checkpointable job to be checkpointed and migrated, even if non-checkpointable conditions exist within the operating system environment.

***job_ID* | "*job_ID[index_list]*" | 0**
Migrates jobs with the specified job IDs. LSF ignores the `-J` and `-u` options.

If you specify a job ID of `0` (zero), LSF ignores all other job IDs and migrates all jobs that satisfy the `-J` and `-u` options.

If you do not specify a job ID, LSF migrates the most recently submitted job that satisfies the `-J` and `-u` options.

-J *job_name*
Migrates the job with the specified name. Ignored if a job ID other than `0` (zero) is specified.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (`*`) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-m "*host_name* ..." | -m "*host_group* ..."
Migrates jobs to the specified hosts.

This option cannot be used on an LSF multicluster capability job. The **bmig** command can restart or rerun only the job on the original host.

-u "*user_name*" | -u "*user_group*" | -u all
Migrates only those jobs that are submitted by the specified users. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

If you specify the reserved user name `all`, LSF migrates jobs that are submitted by all users. Ignored if a job ID other than `0` (zero) is specified.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

bsub, **brestart**, **bchkpnt**, **bjobs**, **bqueues**, **bhosts**, **bugroup**, **mbatchd**, `lsb.queues`, **kill**

Chapter 36. bmod

Modifies job submission options of a job.

Synopsis

bmod [*bsub_options*] [*job_ID* | "*job_ID*[*index*] "]

bmod [-g *job_group_name* | -gn] [*job_ID*]

bmod [-hostfile *file_path* | -hostfilen]

bmod [-sla *service_class_name* | -slan] [*job_ID*]

bmod [-aps "system=*value*" | "admin=*value*" | -apsn] [*job_ID*]

bmod [-h | -V]

Option list

[-ar | -arn]

[-B | -Bn]

[-hl | -hln]

[-N | -Ne | -Nn]

[-r | -rn]

[-ul | -uln]

[-x | -xn]

[-a "esub_application [([*argument*[,*argument*...])]..."]

[-app *application_profile_name* | -appn]

[-aps "system=*value*" | "admin=*value*" | -apsn]

[-b *begin_time* | -bn]

[-C *core_limit* | -Cn]

[-c [*hour*:]*minute*[/*host_name* | /*host_model*] | -cn]

[-clusters "all [~*cluster_name*] ... | *cluster_name*[+[*pref_level*]] ... [others[+[*pref_level*]]]" | -clustern]

[-cwd "*current_working_directory*" | -cwdn]

[-D *data_limit* | -Dn]

[-data "[*host_name*:]*abs_file_path* [[*host_name*:]*abs_file_path* ...]" [-datachk] ... | -data "tag:*tag_name* [tag:*tag_name* ...]" ... | -datan]

[-E "*pre_exec_command* [*argument* ...]" | -En]

[-Ep "*post_exec_command* [*argument* ...]" | -Epn]

[-e *err_file* | -en]

[-eptl [*hour*:]*minute* | -eptln]

[-eo *err_file* | -en]

[-ext[sched] "*external_scheduler_options*"]

[-F *file_limit* | -Fn]

[-f "*local_file op* [*remote_file*]" ... | -fn]

[-freq *numberUnit* | -freqn]

[-G *user_group* | -Gn]

[-g *job_group_name* | -gn]

[-gpu - | "[num=*num_gpus*[/task | host]] [:mode=shared | exclusive_process] [:mps=yes[,shared] | no | per_socket[,shared] | per_gpu[,shared]] [:j_exclusive=yes | no] [:aff=yes | no] [:gmodel=*model_name*[-*mem_size*]] [:gtile=*tile_num*|'!'] [:gmem=*mem_value*] [:nvlink=yes]" | -gpun]

[-i *input_file* | -in | -is *input_file* | -isn]

[-J *job_name* | -J "*%job_limit*" | -Jn]

[-Jd "*job_description*" | -Jdn]

[-k "*checkpoint_dir* [init=*initial_checkpoint_period*] [*checkpoint_period*]" | -kn]

[-L *login_shell* | -Ln]

[-Lp *ls_project_name* | -Lpn]

[-M *mem_limit*[!] | -Mn]

[-m *"host_name*[@*cluster_name*][[!] | +[*pref_level*]] | *host_group*[[!] | +[*pref_level*]| *compute_unit*[[!] | +[*pref_level*]] ..." | -mn]

[-mig *migration_threshold* | -mign]

[-n *min_tasks*[, *max_tasks*] | -nn]

[-network *network_res_req* | -networkn]

[-notify "[exit][done][start][suspend]" | -notifyn]

[-o *out_file* | -on]

[-oo *out_file* | -oon]

[-outdir *output_directory* | -outdirn]

[-P *project_name* | -Pn]

[-p *process_limit* | -pn]

[-ptl [*hour*:]*minute* | -ptln]

[-Q "[*exit_code* ...] [EXCLUDE(*exit_code* ...)]"]

[-q "*queue_name* ..." | -qn]

[-R "*res_req*" [-R "*res_req*" ...] | -Rn]

[-rnc *resize_notification_cmd* | -rncn]

[-S *stack_limit* | -Sn]

[-s *signal* | -sn]

[-sla *service_class_name* | -slan]

[-smt *smt_value* | -smtn]

[-sp *priority* | -spn]

[-stage " [storage= *min_size* [, *max_size*]] [:in=*path_to_stage_in_script*] [:out=*path_to_stage_out_script*] " | -stagen]

[-T *thread_limit* | -Tn]

[-t *term_time* | -tn]

[-ti | -tin]

[-U *reservation_ID* | -Un]

[-u *mail_user* | -un]

[-v *swap_limit* | -vn]

[-W [*hour*:]*minute*[/*host_name* | /*host_model*] | -Wn]

[-We [*hour*:]*minute*[/*host_name* | /*host_model*] | -Wep [*value*] | -We+ [*hour*:]*minute* | -Wen]

[-w "*dependency_expression*" | -wn]

[-wa "[*signal* | *command* | CHKPNT]" | -wan]

[-wt "*job_warning_time*" | -wtn]

[-Z "*new_command*" | -Zs "*new_command*" | -Zsn]

[*job_ID* | "*job_ID*[*index*]"]

Description

Modifies the options of a previously submitted job, including forwarded jobs in an LSF multicluster capability environment. See the **bsub** command reference for complete descriptions of job submission options you can modify with **bmod**.

Only the owner of the job, the user group administrator (for jobs that are associated with a user group), or LSF administrators can modify the options of a job.

All options that are specified at submission time can be changed. The value for each option can be overridden with a new value by specifying the option as in **bsub**. To cancel an option or reset it to its default value, use the option string followed by "n". Do not specify an option value when you reset an option.

The -i, -in, and -Z options have counterparts that support spooling of input and job command files (-is, -isn, -Zs, and -Zsn). Options that are related to file names and job spooling directories support paths that contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

Options that are related to command names can contain up to 4094 characters for UNIX, or up to 255 characters for Windows. Options that are related to job names can contain up to 4094 characters.

You can modify all options of a pending job, even if the corresponding **bsub** command option was not specified.

Modifying options for a forwarded pending job are different from modifying the options of a pending job:

- You cannot modify the following options: -m, -q, -sla, -w.
- For the following options, your modifications take effect only on the submission cluster and not on the execution cluster: -aps, -g, -k.
- For the -J option, if you are changing only the job array limit, the option takes effect only on the submission cluster and not on the execution cluster.
- When you apply a cancellation option (such as the -appn option to cancel a previous -app specification), the default value for that attribute depends on the local cluster configuration.

Modifying a job that is pending in a chunk job queue (the **CHUNK_JOB_SIZE** parameter) removes the job from the chunk to be scheduled later.

Like **bsub**, the **bmod** command calls the master **esub** (**mesub**). The master **esub** runs any mandatory **esub** executable files that are configured by an LSF administrator, and any executable file named **esub** (without *.application* in the file name) if it exists in the LSF_SERVERDIR directory. Only **esub** executable files that are invoked by the **bsub** command can change the job environment on the submission host. An **esub** invoked by the **bmod** command cannot change the job environment. Arguments for **esub** executable files can also be modified.

The -b option modifies the job begin time. If the year field is specified and the specified time is in the past, the start time condition is considered reached and LSF dispatches the job if slots are available.

The -t option modifies job termination time. If the year field is specified and the specified time is in the past, the job modification request is rejected.

Use the -clusters option to modify cluster names for LSF multicluster capability jobs. The bmod -clusters *remote_clusters* command can modify pending jobs only on the submission cluster.

The -cwd option specifies the current working directory (CWD) for a job. The -cwd option works only if the job is in pending state. The path can be absolute or relative to the submission directory. If the submission directory does not exist in the execution host, it tries the logical home directory. If that fails, the tmp directory is used for the CWD.

The path can include the same dynamic patterns as described for the **bsub -cwd** command.

The -cwdn option resets the value for the -cwd option to the submission directory from the **bsub** command.

If the job is submitted with the -app option but without the -cwd option, and the **LSB_JOB_CWD** parameter is not defined, the CWD defined in the specified application profile is used. If CWD is not defined in the application profile, the **DEFAULT_JOB_CWD** value is used. If neither of the two parameters is defined, the submission directory is used for CWD.

The -hl option enables per-job host-based memory and swap limit enforcement on hosts that support Linux cgroups. The -hln option disables host-based memory and swap limit enforcement. The -hl and -hln options apply only to pending jobs. The **LSB_RESOURCE_ENFORCE="memory"** parameter must be specified in the lsf.conf file for host-based memory and swap limit enforcement with the -hl option to take effect. If no memory or swap limit is specified for the job, or the **LSB_RESOURCE_ENFORCE="memory"** parameter is not specified, a host-based memory limit is not set for the job.

The -Epn option cancels the setting of job-level post-execution commands. The job-level post-execution commands do not run. Application-level post-execution commands run if they exist.

If a default user group is configured (with the **DEFAULT_USER_GROUP** parameter in the lsb.params file), the **bmod -Gn** command moves the job to the default user group. If the job is already attached to the default user group, the **bmod -Gn** command has no effect on that job. A job moved to a user group where it cannot run (without shares in a specified fairshare queue, for example) is transferred to the default user group where the job can run.

The -gpu option specifies properties of GPU resources required by the job. For more information on this option see the **bsub -gpu** command.

For resizable jobs, the **bmod -R "rusage[mem | swp]"** command affects the resize allocation request only if the job is not dispatched.

The -M option takes effect only if the job can requeue and when it is run again.

The -m option modifies the first execution host list. When used with a compound resource requirement, the first host that is allocated must satisfy the first simple resource requirement string in the compound resource requirement.

When modifying the notification request using the -notify option, the changed record is also logged to the **JOB_MODIFY2** event. The -notify option does not work with job arrays or job array elements.

The -notifyn option does not work with job array elements.

The -outdir option creates the output directory while the job is in pending state. This option supports the dynamic patterns for the output directory. For example, if user1 runs the command bmod -outdir "/scratch/joboutdir/%U/%J_%I" myjob, the system creates the directory /scratch/joboutdir /user1/jobid_0 for the job output directory.

The -outdirn option resets the output directory value to the **DEFAULT_JOB_OUTDIR** parameter, if it is defined, or sets the output directory to the submission directory where the original **bsub** command ran. The output directory can be modified only while the job is in pending state.

The -Q option does not affect running jobs. For rerunnable and requeue jobs, the -Q option affects the next run.

The -q option resubmits the job to a new queue, as if it was a new submission. By default, LSF dispatches jobs in a queue in order of arrival, so the modified job goes to the last position of the new queue, no matter what its position was in the original queue.

The -rn option resets the rerunnable job setting that is specified by the **bsub -rn** or **bsub -r** option. The application profile and queue level rerunnable job setting if any is used. The **bmod -rn** command does not disable or override job rerun if the job was submitted to a rerunnable queue or application profile with job rerun configured. **bmod -rn** command is different from the **bsub -rn** command, which does override the application profile and queue level rerunnable job setting.

The -ti option enables immediate automatic orphan job termination at the job level. The -ti and -tin options are command flags, not suboptions, so you do not need to respecify the original dependency expression from the -w option that is submitted with the **bsub** command. You can use either the bmod -w [-ti | -tin] or bmod -ti | -tin command. The -tin option cancels the -ti suboption of a submitted dependent job, in which case the cluster-level configuration takes precedence.

The -uln option sets the user shell limits for pending jobs to their default values. The -uln option is not supported on Windows.

-Wen cancels the estimated job run time. The runtime estimate does not take effect for the job.

Modifying running jobs

By default, you can modify resource requirements for running jobs (-R "*res_req*" except -R "cu[*cu_string*]") and the estimated running time for running or suspended jobs (-We, -We+, **-Wep**). To modify more job options for running jobs, define **LSB_MOD_ALL_JOBS=Y** in lsf.conf.

Note: Using the **bmod -R** command with a running job that uses a compound or alternative resource requirement as the effective resource requirement is not permitted. Changing the compound or alternative resource requirement of a running job is also rejected.

When the **LSB_MOD_ALL_JOBS=Y** parameter is set, only some **bsub** options can be modified for running jobs. You cannot make any other modifications after a job is dispatched. You can use **bmod** to modify the following options for running jobs:

- CPU limit (-c [*hour*:]*minute*[/*host_name* | /*host_model*])
- Memory limit (-M *mem_limit*)
- Rerunnable jobs (-r | -rn)
- Resource requirements (-R "*res_req*" except -R "cu[*cu_string*]")
- Run limit (-W *run_limit*[/*host_name* | /*host_model*])

  Note: You can modify the run limit for pending jobs as well.
- Swap limit (-v *swap_limit*)
- Standard output (stdout) file name up to 4094 characters for UNIX and Linux or 255 characters for Windows (-o *output_file*)
- Standard error (stderr) file name up to 4094 characters for UNIX and Linux or 255 characters for Windows (-e *error_file*)
- Overwrite standard output (stdout) file name up to 4094 characters for UNIX and Linux or 255 characters for Windows (-oo *output_file*)
- Overwrite standard error (stderr) file name up to 4094 characters for UNIX and Linux or 255 characters for Windows (-eo *error_file*)

For remote running jobs, you can modify only the following attributes:

- CPU limit ([-c *cpu_limit*[/*host_spec*] | -cn])
- Memory limit ([-M *mem_limit* | -Mn])
- Rerunnable attribute ([-r | -rn])
- Run limit ([-W [*hour*:]*minute*[/*host_name* | /*host_model*] | -Wn])

- Swap limit ([-v *swap_limit* | -vn])
- Standard output/error ([-o *out_file* | -on] [-oo *out_file* | -oon] [-e *err_file* | -en][-eo *err_file* | -en])

Modified resource usage limits cannot exceed limits that are defined in the queue.

To modify the CPU limit or the memory limit of running jobs, the parameters **LSB_JOB_CPULIMIT=Y** and **LSB_JOB_MEMLIMIT=Y** must be defined in the `lsf.conf` file.

By default, options for the following resource usage limits are specified in KB:

- Core limit (`-C`)
- Memory limit (`-M`)
- Stack limit (`-S`)
- Swap limit (`-v`)

Use **LSF_UNIT_FOR_LIMITS** in `lsf.conf` to specify a different unit for the limit (MB, GB, TB, PB, EB, or ZB).

Modifying resource requirements

The `-R` option of **bmod** completely replaces any previous resource requirement specification. It does not add the modification to the existing specification. For example, if you submit a job with

```
bsub -R "rusage[res1=1]"
```

Then modify it with

```
bmod -R "rusage[res2=1]"
```

The new resource usage requirement for the job is `[res2=1]`, not `[res1=1; res2=1]`.

bmod does not support the OR (`||`) operator on the `-R` option.

bmod does not support multiple `-R` option strings for multi-phase `rusage` resource requirements.

Modified `rusage` consumable resource requirements for pending jobs must satisfy any limits set by the parameter **RESRSV_LIMIT** in the `lsb.queues` file. For running jobs, the maximums set by the **RESRSV_LIMIT** parameter must be satisfied but the modified `rusage` values can be lower than the minimum values.

Changes to multi-phase `rusage` strings on running jobs such as `bmod -R "rusage[mem=(mem1 mem2):duration=(dur1 dur2)]"` take effect immediately, and change the remainder of the current phase.

For example, a job is submitted with the following resource requirements:

```
bsub -R "rusage[mem=(500 300 200):duration=(20 30):decay=(1 0)]" myjob
```

And after 15 minutes of run time, the following modification is issued:

```
bmod -R "rusage[mem=(400 300):duration=(20 10):decay=(1 0)]" job_ID
```

The result is the following `rusage` string:

```
rusage[mem=(400 300):duration=(20 10):decay=(1 0)]
```

The running job will reserve `(400-((400-300)*15/20)))=325` MB memory with decay for the next `(20-15)=5` minutes of run time. The second phase then starts, reserving 300 MB of memory for the next 10 minutes with no decay, and end up with no memory that is reserved for the rest of the run time.

After 25 minutes of run time, another modification is issued:

```
bmod -R "rusage[mem=(200 100):duration=(20 10):decay=(1 0)]" job_ID
```

The job reserves 100 MB of memory with no decay for the next 5 minutes of runtime, followed by no reserved memory for the remainder of the job.

To remove all of the string input that is specified by using the **bsub** command, use the -Rn option.

For started jobs, **bmod -R** modifies the effective resource requirements for the job, along with the job resource usage. The effective resource requirement string for scheduled jobs represents the resource requirement that is used by the scheduler to make a dispatch decision. **bmod -R** updates the rusage part in the resource requirements, and keeps the other sections as they were in the original resource requirements. The rusage always represents the job runtime allocation, and is modified along with the job resource usage. For running jobs, you cannot change resource requirements to any of the following resource requirements:

- Compound
- Alternative
- rusage siblings
- Compound to simple

Modifying the estimated run time of jobs

The following options modify the estimated run time of a job:

-We [*hour*:]*minute*[/*host_name* | /*host_model*]
Sets an estimated run time. Specifying a host or host model normalizes the time with the CPU factor (time/CPU factor) of the host or model.

-We+ [*hour*:]*minute*]
Sets an estimated run time that is the value you specify added to the accumulated run time. For example, if you specify -We+ 30 and the job runs for 60 minutes, the new estimated run time is now 90 minutes.

Specifying a host or host model normalizes the time with the CPU factor (time/CPU factor) of the host or model.

-Wep [*value*]
Sets an estimated run time that is the percentage of job completion that you specify added to the accumulated run time. For example, if you specify -Wep+ 25 (meaning that the job is 25% complete) and the job runs for 60 minutes, the new estimated run time is now 240 minutes.

The range of valid values is greater than 0 and less than or equal to 100. Two digits after decimal are supported.

Specifying a host or host model normalizes the time with the CPU factor of the host or model (time/CPU factor).

Modifying job groups

Use the -g option of **bmod** and specify a job group path to move a job or a job array from one job group to another. For example, the following command moves job 105 to job group /risk_group/portfolio2/monthly:

```
bmod -g /risk_group/portfolio2/monthly 105
```

Like **bsub -g**, if the job group does not exist, LSF creates it.

The **bmod -g** option cannot be combined with other **bmod** options. It can operate only on pending jobs. It cannot operate on running or finished jobs.

You can modify your own job groups and job groups that other users create under your job groups. LSF administrators can modify job groups of all users.

You cannot move job array elements from one job group to another, only entire job arrays. If any job array elements in a job array are running, you cannot move the job array to another group. A job array can belong only to one job group at a time.

You cannot modify the job group of a job that is attached to a service class. Job groups cannot be used with resource-based SLAs that have guarantee goals.

If you want to specify array dependency by array name, set the **JOB_DEP_LAST_SUB** parameter in the `lsb.params` file. If this parameter is not set, the job is rejected if one of your previous arrays has the same name but a different index.

Modifying jobs in service classes

The **-sla** option modifies a job by attaching it to the specified service class. The **-slan** option detaches the specified job from a service class. If the service class does not exist, the job is not modified. For example, the following command attaches job 2307 to the service class `Duncan`:

```
bmod -sla Duncan 2307
```

The following command detaches job 2307 from the service class `Duncan`. If a default SLA is configured in `lsb.params`, the job is moved to the default service class.

```
bmod -slan 2307
```

You cannot do the following job modifications:

- Use the `-sla` option with other **bmod** options
- Modify the service class of job that is already attached to a job group. (Time-based SLAs only.) Use the **bsla** command to display the configuration properties of service classes that are configured in the `lsb.serviceclasses` file, and dynamic information about the state of each service class.
- Modify a job such that it no longer satisfies the assigned guarantee SLA. Jobs auto-attached to guarantee SLAs reattach to another SLA as required, but jobs that are submitted with an SLA specified must continue to satisfy the SLA access restrictions.

If a default SLA is configured with the **ENABLE_EGO_DEFAULT_SLA** parameter in the `lsb.params` file, the **bmod -slan** option moves the job to the default SLA. If the job is already attached to the default SLA, the **bmod -slan** option has no effect on that job.

Modifying jobs associated with application profiles

The `-app` option modifies a job by associating it to the specified application profile. The `-appn` option dissociates the specified job from its application profile. If the application profile does not exist, the job is not modified.

You can modify the application profile only for pending jobs. For example, the following command associates job 2308 with the application profile `fluent`:

```
bmod -app fluent 2308
```

The following command dissociates job 2308 from the service class `fluent`:

```
bmod -appn 2308
```

Use **bapp** to display the properties of application profiles that are configured in `LSB_CONFDIR/`*`cluster_name`*`/configdir/lsb.applications`.

Modifying absolute priority scheduling options

Administrators can use **bmod -aps** to adjust the APS value for pending jobs. **bmod -apsn** cancels previous **bmod -aps** settings. You cannot combine **bmod -aps** with other **bmod** options.

You can change the APS value only for pending resizable jobs.

-aps "system=*value***"** ***job_ID***
: Set a static nonzero APS value of a pending job. Setting a system APS value overrides any calculated APS value for the job. The system APS value cannot be applied to running jobs.

-aps "admin=*value*" *job_ID*
Set a nonzero **ADMIN** factor value for a pending job. The **ADMIN** factor adjusts the calculated APS value higher or lower. A negative admin value lowers the calculated APS value, and a positive value raises the calculated APS value relative to other pending jobs in the APS queue.

You cannot configure APS weight, limit, or grace period for the ADMIN factor. The **ADMIN** factor takes effect as soon as it is set.

-apsn
Use the **bmod -apsn** option to cancel previous **bmod -aps** settings. You cannot apply the **bmod -apsn** option to running jobs in an APS queue. An error is issued if the job has no system APS priority or **ADMIN** factor set.

Modifying resizable jobs

Use the `-rnc` and `-ar` options to modify the autoresizable attribute or resize notification command for resizable jobs. You can modify the autoresizable attribute only for pending jobs (PSUSP or PEND). You can modify the resize notification command only for unfinished jobs (not DONE or EXIT jobs).

-rnc *resize_notification_cmd*
Specify the name of an executable file to be invoked on the first execution host when the job allocation is modified (both shrink and grow). The **bmod -rnc** option overrides any notification command that is specified in the application profile.

-rncn
Remove the notification command setting from the job.

-ar
Specify that the job is autoresizable.

-arn
Remove job-level autoresizable attribute from the job.

Modifying network scheduling options for PE jobs

The `-network` option modifies the network scheduling options for IBM Parallel Environment (PE) jobs. The `-networkn` option removes any network scheduling options for the PE job.

You cannot modify the network scheduling options for running jobs, even if `LSB_MOD_ALL_JOBS=y`.

Modifying memory rusage for affinity jobs

When you use the **bmod** command to modify memory `rusage` of a running job with an affinity resource request, host-level available memory and available memory in NUMA nodes might be inconsistent when you use **bhosts -l -a**. Inconsistencies happen because the modified resource requirement takes effect in the next scheduling cycle for affinity scheduling, but it takes effect immediately at the host level. The **bmod** command updates only resource usage that LSF accounts; it has no effect on the running jobs. For memory binding, when a process is bound to some NUMA node, LSF limits which NUMA node the process gets physical memory from. LSF does not ask the operating system to reserve any physical memory for the process.

Modifying jobs with a user-specified host file

Use the `-hostfile` option to modify a pending job with a user-specified host file.

```
bmod -hostfile "host_alloc_file" ./a.out <job_id>
```

A user-specified host file contains specific hosts and slots that a user wants to use for a job. For example, if you know what the best host allocation for a job is based on factors such as network connection status, you can choose to submit a job with a user specified host file. The user specified host file specifies the order in which to launch tasks, ranking the slots specified in the file. The resulting rank file is also made available to other applications (such as MPI).

Important:

- The `-hostfile` cannot be used with either the –n or –m option.
- The `-hostfile` option cannot be combined with –R or compound *res_req*.
- Do not use a user specified host file if you enabled task geometry because it can cause conflicts and jobs might fail.
- If resources are not available at the time that a task is ready, use advance reservation instead of a user-specified host file, to ensure that reserved slots are available and to guarantee that a job runs smoothly.

Any user can create a user specified host file. It must be accessible by the user from the submission host. It lists one host per line in the following format:

```
# This is a user specified host file
<host_name1>   [<# slots>]
<host_name2>   [<# slots>]
<host_name1>   [<# slots>]
<host_name2>   [<# slots>]
<host_name3>   [<# slots>]
<host_name4>   [<# slots>]
```

The following rules apply to the user specified host file:

- Specifying the number of slots for a host is optional. If no slot number is indicated, the default is 1.
- A host name can be either a host in a local cluster or a host leased-in from a remote cluster (*host_name*@*cluster_name*).
- A user specified host file must contain hosts from the same cluster only.
- A host name can be entered with or without the domain name.
- Host names can be used multiple times, and the order of the hosts represents the placement of tasks.

```
#first three tasks
host01                      3
#fourth tasks
host02
#next three tasks
host03                      3
```

- Start comments with the # character.

The user specified host file is deleted along with other job-related files when a job is cleaned.

To remove a user-specified host file that is specified for a pending job, use the `-hostfilen` option:

```
bmod -hostfilen <job_id>
```

Modifying job data requirements

The `-data` option modifies the data staging requirements for a pending job that is submitted with the `bsub -data` option. If file transfers for previously specified files are still in progress, they are not stopped. Only new transfers are initiated for the new data management requirement as needed. Use `-datan` to cancel the data staging requirement for the job.

Modifying the data requirements of a job replaces the entire previous data requirement string in the `-data` option with the new one. When you modify or add a part of the data requirement string, you must specify the entire data requirement string in **bmod -data** with the modifications.

The sanity check for the existence of files or folders and whether the user can access them, discovery of the size and modification time of the files or folders, and generation of the hash from the **bmod** command occurs in the transfer job. This equalizes modification performance between jobs with and without data requirements. The `-datachk` option can perform full checking for jobs with a data requirement. The `-datachk` option can be specified only with the **-data** command. If the data requirement is for a tag, this option has no effect.

You must have read access to the specified file to modify the data requirements for the job.

You cannot use **bmod -data** or **bmod -datan** to modify the data management requirements of a running or finished job, or for a job that is already forwarded to or from another cluster.

For example, if a job that was originally submitted with the following command:

```
bsub -data /proj/user1/dataset_A_%I -J A[1-10] myjob.sh
```

And you modify the job with the following command:

```
bmod -data /proj/user1/dataset_B_%I "A[1]"
```

Then all the paths /proj/user1/dataset_B_1, /proj/user1/dataset_B_2 ... /proj/user1/dataset_B_10 must exist. If the **bmod** command succeeds, then A[1] still has all 10 data files in its data requirement.

Options

***job_ID* | "*job_ID*[*index*]"**
Modifies jobs with the specified job ID.

Modifies job array elements that are specified by "*job_ID*[*index*]".

-h
Prints command usage to stderr and exits.

-V
Prints LSF release version to stderr and exits.

Limitations

If you do not specify -e or -eo before the job is dispatched, you cannot modify the name of job error file for a running job. Modifying the job output options of remote running jobs is not supported.

See also

bsub

Chapter 37. bparams

Displays information about configurable system parameters in the `lsb.params` file.

Synopsis

`bparams [-a] [-l]`

`bparams [-h | -V]`

Description

Displays the following parameter values:

- Default queues
- The value of the **MBD_SLEEP_TIME** parameter, which is used for calculations
- Job checking interval
- Job accepting interval

Options

-a
All format. Displays all the configurable parameters set in the `lsb.params` file.

-l
Long format. Displays detailed information about all the configurable parameters in the `lsb.params` file.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

`lsb.params`

Chapter 38. bpeek

Displays the `stdout` and `stderr` output of an unfinished job.

Synopsis

bpeek [`-f`] [`-q` *queue_name* | `-m` *host_name* | `-J` *job_name* | *job_ID* | "*job_ID*[*index_list*]"]

bpeek [`-h` | `-V`]

Description

Displays the standard output and standard error output that is produced by one of your unfinished jobs, up to the time that you run the command.

By default, displays the output by using the command **`cat`**.

This command is useful for monitoring the progress of a job and identifying errors. If errors are observed, valuable user time and system resources can be saved by stopping an erroneous job.

Options

-f
Displays the output of the job by using the command **`tail -f`**. When the job is completed, the **bpeek `-f`** command exits.

If the peeked job is requeued or migrated, abort any previous **bpeek `-f`** command and rerun the **bpeek `-f`** command.

-q *queue_name*
Operates on your most recently submitted job in the specified queue.

-m *host_name*
Operates on your most recently submitted job that was dispatched to the specified host.

-J *job_name*
Operates on your most recently submitted job that has the specified job name.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (`*`) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

***job_ID* | "*job_ID[index_list]*"**
Operates on the specified job.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

`cat`, **`tail`**, **`bsub`**, **`bjobs`**, **`bhist`**, **`bhosts`**, **`bqueues`**

Chapter 39. bpost

Sends external status messages and attaches data files to a job.

Synopsis

bpost [-i *message_index*] [-d "*description*" [-N INFO | WARNING | ERROR | CRITICAL]] [-a *data_file*] *job_ID* | "*job_ID*[*index*]" | -J *job_name*

bpost [-h | -V]

Description

Provides external status information or sends data to a job in the system.

By default, operates on the message index 0. By default, posts the message "no description".

If you specify a job ID, the **bpost** command has the following restrictions:

- You can send messages and data only to your own jobs.
- You cannot send messages and data to jobs submitted by other users.
- Only root and LSF administrators can send messages to jobs submitted by other users.
- Root and LSF administrators cannot attach data files to jobs submitted by other users.

Job names are not unique. If you specify the -J *job_name* option, the **bpost** command has the following restrictions:

- You can send messages and data only to your own jobs.
- You cannot send messages and data to jobs submitted by other users.
- Root and the LSF administrators can send messages and data only to their own jobs.

A job can accept messages until it is cleaned from the system. If your application requires transfer of data from one job to another, use the -a option of the **bpost** command to attach a data file to the job. Then, use the **bread** command to copy the attachment to another file.

You can associate several messages and attached data files with the same job. As the job is processed, use the **bread** or **bstatus** command to retrieve the messages posted to the job. Use the **bread** to copy message attachments to external files.

For example, your application might require job status descriptions besides the ones that LSF provides internally (PEND, RUN, SUSP). Use the -d option to place your own status or job description text as a message to the job.

You can also use **bstatus -d** command to update the external job status. The following command

```
bstatus -d "description" myjob
```

Is equivalent to the following command:

```
bpost -i 0 -d "description" myjob
```

With LSF multicluster capability, both clusters must run LSF Version 7 or later. You cannot attach files to LSF multicluster capability jobs.

If the LSF multicluster capability connection is lost (the **mbatchd** daemon is down), messages can be saved and resent when the connection is recovered. The messages are backed up on the local cluster and resent in their original order.

Options

-a *data_file*
Attaches the specified data file to the job external storage. This option is ignored for LSF multicluster capability jobs; you can attach a file only if the job runs in the local cluster.

Use the **JOB_ATTA_DIR** parameter in the `lsb.params` file to specify the directory where attachment data files are saved. The directory must have at least 1 MB of free space. The **mbatchd** daemon checks for available space in the job attachment directory before it transfers the file.

Use the **MAX_JOB_ATTA_SIZE** parameter in the `lsb.params` file to set a maximum size for job message attachments.

-d "*description*"
Places your own status text as a message to the job. The message description has a maximum length of 512 characters.

For example, your application might require job status descriptions besides the ones that LSF provides internally (PEND, RUN, SUSP).

The default description is `"no description"`.

-i *message_index*
Operates on the specified message index.

The default message index is `0`.

Use the **MAX_JOB_MSG_NUM** parameter in the `lsb.params` file to set a maximum number of messages for a job. With LSF multicluster capability, to avoid conflicts, the **MAX_JOB_MSG_NUM** parameter must be the same in all clusters.

-N INFO | WARNING | ERROR | CRITICAL
Sends a message (from the `-d` option) at the specified notification level to LSF Application Center Notifications as specified by the **LSF_AC_PNC_URL** parameter in the `lsf.conf` file. The length of a notification must be less than 1024 bytes, otherwise LSF truncates the message.

job_ID* | "*job_ID[index]*" | -J *job_name
Required. Operates on the specified job. With LSF multicluster capability job forwarding model, you must always use the local job ID.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (`*`) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Example

```
bpost -i 1 -N INFO -d "step 1" -a step1.out 2500
```

Puts the message text `step 1` into message index 1 and LSF Application Center Notifications at the `INFO` level (if the **LSF_AC_PNC_URL** parameter is specified correctly in the `lsf.conf` file), and attaches the file `step1.out` to job 2500.

See also

bread, **bstatus**, **MAX_JOB_ATTA_SIZE**, **MAX_JOB_MSG_NUM**

Chapter 40. bqueues

Displays information about queues.

Synopsis

bqueues [-w | -l | -r | -o "*field_name*[:[-][*output_width*]] ... [delimiter='*character*']" [-json]] [-m *host_name* | -m *host_group* | -m *cluster_name* / -m all] [-u *user_name* | -u *user_group* / -u all] [*queue_name* ...] [-alloc]

bqueues [-w | -o "*field_name*[:[-][*output_width*]] ... [delimiter='*character*']"] [-m *host_name* | -m *host_group* | -m *cluster_name* / -m all] [-u *user_name* | -u *user_group* / -u all] [*queue_name* ...] [-alloc] [-noheader]

bqueues [-h | -V]

Description

By default, returns the following information about all queues: queue name, queue priority, queue status, task statistics, and job state statistics.

When a resizable job has a resize allocation request, **bqueues** displays pending requests. When LSF adds more resources to a running resizable job, **bqueues** decreases job PEND counts and displays the added resources. When LSF removes resources from a running resizable job, **bqueues** displays the updated resources.

In LSF multicluster capability, returns the information about all queues in the local cluster.

Returns job slot statistics if the -alloc option is used.

Batch queue names and characteristics are set up by the LSF administrator in the lsb.queues file.

CPU time is normalized.

CPU time output is not consistent with the bacct command

The **bacct** command displays the sum of CPU time that is used by all past jobs in event files. If you specify a begin and end time, the execution host type and run time are also considered in the CPU time. For a specified job, the **bacct** and **bhist** commands have the same result.

Because the value of CPU time for the **bqueues** command is used by the **mbatchd** daemon to calculate fairshare priority, it does not display the actual CPU time for the queue. CPU time is normalized by CPU factor. Normalized CPU time results in a different CPU time output in the **bacct** and **bqueues** commands.

Options

-alloc
Shows counters for slots in RUN, SSUSP, USUSP, and RSV. The slot allocation is different depending on whether the job is an exclusive job or not.

-json

Displays the customized output in JSON format.

When specified, **bqueues -o** displays the customized output in the JSON format.

This option only applies to output for the **bqueues -o** command for customized output. This has no effect when running **bqueues** without the -o option and the **LSB_BQUEUES_FORMAT** environment variable or parameter are not defined.

-l

Displays queue information in a long multiline format. The -l option displays the following additional information:

- Queue description
- Queue characteristics and statistics
- Scheduling parameters
- Resource usage limits
- Scheduling policies
- Users
- Hosts
- Associated commands
- Dispatch and run windows
- Success exit values
- Host limits per parallel job
- Pending time limits and eligible pending time limits
- Job controls
- User shares
- Normalized fairshare factors
- Containers

If you specified an administrator comment with the `-C` option of the queue control commands (**qclose**, **qopen**, **qact**, and **qinact**), **qhist** displays the comment text.

Displays absolute priority scheduling (APS) information for queues that are configured with the **APS_PRIORITY** parameter.

-noheader

Removes the column headings from the output.

When specified, **bqueues** displays the values of the fields without displaying the names of the fields. This is useful for script parsing, when column headings are not necessary.

This option applies to output for the **bqueues** command with no options, and to output for all **bqueues** options with output that uses column headings, including the following: `-alloc`, `-m`, `-o`, `-u`, `-w`.

This option does not apply to output for **bqueues** options that do not use column headings, including the following: `-json`, `-l`, `-r`.

-o

Sets the customized output format.

- Specify which **bqueues** fields (or aliases instead of the full field names), in which order, and with what width to display.
- Specify only the **bqueues** field name or alias to set its output to unlimited width and left justification.
- Specify the colon (`:`) without a width to set the output width to the recommended width for that field.
- Specify the colon (`:`) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **bqueues** truncates the ending characters.
- Specify a hyphen (`-`) to set right justification when **bqueues** displays the output for the specific field. If not specified, the default is to set left justification when **bqueues** displays output for a field.
- Use `delimiter=` to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the output for certain **bqueues** options:

- **LSB_BQUEUES_FORMAT** and **bqueues -o** both apply to output for the **bqueues** command with no options, and for **bqueues** options with output that filter information, including the following options: `-alloc`, `-m`, `-u`.
- **LSB_BQUEUES_FORMAT** and **bqueues -o** do not apply to output for **bqueues** options that use a modified format, including the following options: `-l`, `-r`, `-w`.

The **bqueues -o** option overrides the **LSB_BQUEUES_FORMAT** environment variable, which overrides the **LSB_BQUEUES_FORMAT** setting in `lsf.conf`.

The following are the field names used to specify the **bqueues** fields to display, recommended width, aliases you can use instead of field names, and units of measurement for the displayed field:

Table 5. Output fields for bqueues

Field name	Width	Aliases	Unit
queue_name	15	qname	
description	50	desc	
priority	10	prio	
status	12	stat	
max	10		
jl_u	10	jlu	
jl_p	10	jlp	
jl_h	10	jlh	
njobs	10		
pend	10		
run	10		
susp	10		
rsv	10		
ususp	10		
ssusp	10		
nice	6		

Field names and aliases are not case-sensitive. Valid values for the output width are any positive integer 1 - 4096.

For example,

```
bqueues -o "queue_name description:10 priority:- status: max:-6
delimiter='^'"
```

This command displays the following fields:

- QUEUE_NAME with unlimited width and left justified.
- DESCRIPTION with a maximum width of ten characters and left justified.
- PRIORITY with a maximum width of ten characters (which is the recommended width) and right justified.
- STATUS with a maximum width of 12 characters (which is the recommended width) and left justified.
- MAX with a maximum width of six characters and right justified.
- The ^ character is displayed between different headers and fields.

-r

Displays the same information as the `-l` option. In addition, if fairshare is defined for the queue, displays recursively the share account tree of the fairshare queue. When queue-based fairshare is used along with the **bsub -G** command and the **LSB_SACCT_ONE_UG=Y** parameter in the `lsf.conf` file, share accounts are only created for active users and for the default user group (if defined).

Displays the global fairshare policy name for the participating queue. Displays remote share load (`REMOTE_LOAD` column) for each share account in the queue.

Displays the normalized fairshare factor, if it is not zero.

-w

Displays queue information in a wide format. Fields are displayed without truncation.

-m *host_name* | -m *host_group* | -m *cluster_name* | -m all

Displays the queues that can run jobs on the specified host. If the keyword `all` is specified, displays the queues that can run jobs on all hosts.

If a host group is specified, displays the queues that include that group in their configuration. For a list of host groups, use the **bmgroup** command.

In LSF multicluster capability, if the `all` keyword is specified, displays the queues that can run jobs on all hosts in the local cluster. If a cluster name is specified, displays all queues in the specified cluster.

-u *user_name* | -u *user_group* | -u all

Displays the queues that can accept jobs from the specified user. If the keyword `all` is specified, displays the queues that can accept jobs from all users.

If a user group is specified, displays the queues that include that group in their configuration. For a list of user groups, use the **bugroup** command.

***queue_name* ...**

Displays information about the specified queues.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Default Output

Displays the following fields:

QUEUE_NAME
The name of the queue. Queues are named to correspond to the type of jobs that are usually submitted to them, or to the type of services they provide.

lost_and_found
If the LSF administrator removes queues from the system, LSF creates a queue that is called `lost_and_found` and places the jobs from the removed queues into the `lost_and_found` queue. Jobs in the `lost_and_found` queue are not started unless they are switched to other queues the **bswitch** command.

PRIO
The priority of the queue. The larger the value, the higher the priority. If job priority is not configured, determines the queue search order at job dispatch, suspend, and resume time. Contrary to usual order of UNIX process priority, jobs from higher priority queues are dispatched first and jobs from lower priority queues are suspended first when hosts are overloaded.

STATUS
The status of the queue. The following values are supported:

Open
The queue can accept jobs.

Closed
The queue cannot accept jobs.

Active
Jobs in the queue can be started.

Inactive
Jobs in the queue cannot be started.

At any moment, each queue is in either `Open` or `Closed` state, and is in either `Active` or `Inactive` state. The queue can be opened, closed, inactivated, and reactivated with the **badmin** command.

Jobs that are submitted to a queue that is later closed are still dispatched while the queue is active. The queue can also become inactive when either its dispatch window is closed or its run window is closed. In this case, the queue cannot be activated by using **badmin**. The queue is reactivated by LSF when one of its dispatch windows and one of its run windows are open again. The initial state of a queue at LSF startup `Open`, and either `Active` or `Inactive` depending on its dispatch windows.

MAX
The maximum number of job slots that can be used by the jobs from the queue. These job slots are used by dispatched jobs that are not yet finished, and by pending jobs that reserve slots.

A sequential job uses one job slot when it is dispatched to a host, while a parallel job uses as many job slots as is required by **bsub -n** command when it is dispatched. A dash (-) indicates no limit.

JL/U
The maximum number of job slots each user can use for jobs in the queue. These job slots are used by your dispatched jobs that are not yet finished, and by pending jobs that reserve slots. A dash (-) indicates no limit.

JL/P
The maximum number of job slots a processor can process from the queue. This number includes job slots of dispatched jobs that are not yet finished, and job slots reserved for some pending jobs. The job slot limit per processor controls the number of jobs that are sent to each host. This limit is configured per processor so that multiprocessor hosts are automatically allowed to run more jobs. A dash (-) indicates no limit.

JL/H

The maximum number of job slots a host can allocate from this queue. This number includes the job slots of dispatched jobs that are not yet finished, and slots that are reserved for some pending jobs. The job slot limit per host (JL/H) controls the number of jobs that are sent to each host, regardless of whether a host is a uniprocessor host or a multiprocessor host. A dash (-) indicates no limit.

NJOBS
The total number of slots for jobs in the queue. This number includes slots for pending, running, and suspended jobs. Batch job states are described in the **bjobs** command.

If the `-alloc` option is used, the total is the sum of the RUN, SSUSP, USUSP, and RSV counters.

PEND
The total number of tasks for all pending jobs in the queue. If used with the `-alloc` option, total is zero.

RUN
The total number of tasks for all running jobs in the queue. If the `-alloc` option is used, the total is allocated slots for the jobs in the queue.

SUSP
The total number of tasks for all suspended jobs in the queue.

PJOBS
The total number of pending jobs (including both PEND and PSUSP job) in this queue

Long Output (-l)

In addition to the default fields, the `-l` option displays the following fields:

Description
A description of the typical use of the queue.

Default queue indication
Indicates the default queue.

PARAMETERS/ STATISTICS

NICE
The UNIX nice value at which jobs in the queue are run. The nice value reduces process priority.

STATUS

Inactive
The long format for the `-l` option gives the possible reasons for a queue to be inactive:

Inact_Win
The queue is out of its dispatch window or its run window.

Inact_Adm
The queue is inactivated by the LSF administrator.

SSUSP

The number of tasks for all jobs in the queue that are suspended by LSF because of load levels or run windows. If `-alloc` is used, the total is the allocated slots for the jobs in the queue.

USUSP

The number of tasks for all jobs in the queue that are suspended by the job submitter or by the LSF administrator. If `-alloc` is used, the total is the allocated slots for the jobs in the queue.

RSV

For pending jobs in the queue, the number of tasks that LSF reserves slots for. If `-alloc` is used, the total is the allocated slots for the jobs in the queue.

Migration threshold
The length of time in seconds that a job that is dispatched from the queue remains suspended by the system before LSF attempts to migrate the job to another host. See the **MIG** parameter in the `lsb.queues` and `lsb.hosts` files.

Schedule delay for a new job
The delay time in seconds for scheduling after a new job is submitted. If the schedule delay time is zero, a new scheduling session is started as soon as the job is submitted to the queue. See the **NEW_JOB_SCHED_DELAY** parameter in the `lsb.queues` file.

Interval for a host to accept two jobs
The length of time in seconds to wait after a job is dispatched to a host and before a second job is dispatched to the same host. If the job accept interval is zero, a host can accept more than one job in each dispatching interval. See the **JOB_ACCEPT_INTERVAL** parameter in the `lsb.queues` and `lsb.params` files.

RESOURCE LIMITS
The hard resource usage limits that are imposed on the jobs in the queue (see **getrlimit** and the `lsb.queues` file). These limits are imposed on a per-job and a per-process basis.

The following per-job limits are supported:

CPULIMIT
The maximum CPU time a job can use, in minutes, relative to the CPU factor of the named host. `CPULIMIT` is scaled by the CPU factor of the execution host so that jobs are allowed more time on slower hosts.

When the job-level CPULIMIT is reached, a SIGXCPU signal is sent to all processes that belong to the job. If the job has no signal handler for SIGXCPU, the job is killed immediately. If the SIGXCPU signal is handled, blocked, or ignored by the application, then after the grace period expires, LSF sends SIGINT, SIGTERM, and SIGKILL signals to the job to kill it.

TASKLIMIT
The maximum number of tasks that are allocated to a job. Jobs that have fewer tasks than the minimum TASKLIMIT or more tasks than the maximum TASKLIMIT are rejected. Maximum tasks that are requested cannot be less than the minimum TASKLIMIT, and minimum tasks that are requested cannot be more than the maximum TASKLIMIT.

MEMLIMIT
The maximum running set size (RSS) of a process. If a process uses more memory than the limit allows, its priority is reduced so that other processes are more likely to be paged in to available memory. This limit is enforced by the **setrlimit** system call if it supports the RLIMIT_RSS option.

By default, the limit is shown in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the lsf.conf file to specify a larger unit for display (MB, GB, TB, PB, or EB).

SWAPLIMIT
The swap space limit that a job can use. If SWAPLIMIT is reached, the system sends the following signals in sequence to all processes in the job: SIGINT, SIGTERM, and SIGKILL.

By default, the limit is shown in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the lsf.conf file to specify a larger unit for display (MB, GB, TB, PB, or EB).

PROCESSLIMIT
The maximum number of concurrent processes that are allocated to a job. If PROCESSLIMIT is reached, the system sends the following signals in sequence to all processes that belong to the job: SIGINT, SIGTERM, and SIGKILL.

THREADLIMIT
The maximum number of concurrent threads that are allocated to a job. If THREADLIMIT is reached, the system sends the following signals in sequence to all processes that belong to the job: SIGINT, SIGTERM, and SIGKILL.

RUNLIMIT
The maximum wall clock time a process can use, in minutes. RUNLIMIT is scaled by the CPU factor of the execution host. When a job is in RUN state for a total of RUNLIMIT minutes, LSF sends a SIGUSR2 signal to the job. If the job does not exit within 10 minutes, LSF sends a SIGKILL signal to kill the job.

FILELIMIT
The maximum file size a process can create, in KB. This limit is enforced by the UNIX **setrlimit** system call if it supports the RLIMIT_FSIZE option, or the **ulimit** system call if it supports the UL_SETFSIZE option.

DATALIMIT
The maximum size of the data segment of a process, in KB. The data limit restricts the amount of memory a process can allocate. DATALIMIT is enforced by the **setrlimit** system call if it supports the RLIMIT_DATA option, and unsupported otherwise.

STACKLIMIT
The maximum size of the stack segment of a process. This limit restricts the amount of memory a process can use for local variables or recursive function calls. STACKLIMIT is enforced by the **setrlimit** system call if it supports the RLIMIT_STACK option.

By default, the limit is shown in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the lsf.conf file to specify a larger unit for display (MB, GB, TB, PB, or EB).

CORELIMIT
The maximum size of a core file. This limit is enforced by the **setrlimit** system call if it supports the RLIMIT_CORE option.

If a job submitted to the queue specifies any of these limits, then the lower of the corresponding job limits and queue limits are used for the job.

If no resource limit is specified, the resource is assumed to be unlimited.

By default, the limit is shown in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for display (MB, GB, TB, PB, or EB).

HOSTLIMIT_PER_JOB

The maximum number of hosts that a job in this queue can use. LSF verifies the host limit during the allocation phase of scheduling. If the number of hosts that are requested for a parallel job exceeds this limit and LSF cannot satisfy the minimum number of request slots, the parallel job pends.

SCHEDULING PARAMETERS

The scheduling and suspending thresholds for the queue.

The scheduling threshold `loadSched` and the suspending threshold `loadStop` are used to control batch job dispatch, suspension, and resumption. The queue thresholds are used in combination with the thresholds that are defined for hosts. If both queue level and host level thresholds are configured, the most restrictive thresholds are applied.

The `loadSched` and `loadStop` thresholds have the following fields:

r15s

The 15 second exponentially averaged effective CPU run queue length.

r1m

The 1 minute exponentially averaged effective CPU run queue length.

r15m

The 15 minute exponentially averaged effective CPU run queue length.

ut

The CPU usage exponentially averaged over the last minute, expressed as a percentage between 0 and 1.

pg

The memory paging rate exponentially averaged over the last minute, in pages per second.

io

The disk I/O rate exponentially averaged over the last minute, in KB per second.

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard has not been touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver becomes active on a particular host.

tmp

The amount of free space in `/tmp`, in MB.

swp

The amount of currently available swap space. By default, swap space is shown in MB. Use the **LSF_UNIT_FOR_LIMITS** in `lsf.conf` to specify a different unit for display (KB, MB, GB, TB, PB, or EB).

mem

The amount of currently available memory. By default, memory is shown in MB. Use the **LSF_UNIT_FOR_LIMITS** in `lsf.conf` to specify a different unit for display (KB, MB, GB, TB, PB, or EB).

cpuspeed

The speed of each individual cpu, in megahertz (MHz).

bandwidth

The maximum bandwidth requirement, in megabits per second (Mbps).

In addition to these internal indices, external indices are also displayed if they are defined in `lsb.queues` (see `lsb.queues`(5)).

The `loadSched` threshold values specify the job dispatch thresholds for the corresponding load indices. If a dash (-) is displayed as the value, it means that the threshold is not applicable. Jobs in the queue might be dispatched to a host if the values of all the load indices of the host are within the corresponding thresholds of the queue and the host. Load indices can be below or above the threshold, depending on the meaning of the load index. The same conditions are used to resume jobs that are dispatched from the queue that are suspended on this host.

Similarly, the `loadStop` threshold values specify the thresholds for job suspension. If any of the load index values on a host go beyond the corresponding threshold of the queue, jobs in the queue are suspended.

JOB EXCEPTION PARAMETERS

Configured job exception thresholds and number of jobs in each exception state for the queue.

`Threshold` and `NumOfJobs` have the following fields:

overrun

Configured threshold in minutes for overrun jobs, and the number of jobs in the queue that triggered an overrun job exception by running longer than the overrun threshold.

underrun

Configured threshold in minutes for underrun jobs, and the number of jobs in the queue that triggered an underrun job exception by finishing sooner than the underrun threshold.

idle

Configured threshold (CPU time/runtime) for idle jobs, and the number of jobs in the queue that triggered an overrun job exception by having a job idle factor less than the threshold.

SCHEDULING POLICIES

Scheduling policies of the queue. Optionally, one or more of the following policies can be configured in the `lsb.queues` file:

APS_PRIORITY

Absolute Priority Scheduling is enabled. Pending jobs in the queue are ordered according to the calculated APS value.

FAIRSHARE

Queue-level fairshare scheduling is enabled. Jobs in this queue are scheduled based on a fairshare policy instead of the first-come, first-served (FCFS) policy.

BACKFILL

A job in a backfill queue can use the slots that are reserved by other jobs if the job can run to completion before the slot-reserving jobs start.

Backfilling does not occur on queue limits and user limit but only on host-based limits. That is, backfilling is only supported when `MXJ`, `JL/U`, `JL/P`, `PJOB_LIMIT`, and `HJOB_LIMIT` limits are reached. Backfilling is not supported when `MAX_JOBS`, `QJOB_LIMIT`, and `UJOB_LIMIT` are reached.

IGNORE_DEADLINE
If the **IGNORE_DEADLINE=Y** parameter is set in the queue, starts all jobs regardless of the run limit.

EXCLUSIVE
Jobs that are dispatched from an exclusive queue can run exclusively on a host if the user so specifies at job submission time. Exclusive execution means that the job is sent to a host with no other running batch jobs. No further jobs are dispatched to that host while the job is running. The default is not to allow exclusive jobs.

NO_INTERACTIVE
This queue does not accept batch interactive jobs that are submitted with the `-I`, `-Is`, and `-Ip` options of the **bsub** command. The default is to accept both interactive and non-interactive jobs.

ONLY_INTERACTIVE
This queue accepts only batch interactive jobs. Jobs must be submitted with the `-I`, `-Is`, and `-Ip` options of the **bsub** command. The default is to accept both interactive and non-interactive jobs.

SLA_GUARANTEES_IGNORE
This queue is allowed to ignore SLA resource guarantees when scheduling jobs.

FAIRSHARE_QUEUES
Lists queues that participate in cross-queue fairshare. The first queue that is listed is the master queue, which is the queue where fairshare is configured. All other queues that are listed inherit the fairshare policy from the master queue. Fairshare information applies to all the jobs that are running in all the queues in the fair share tree.

QUEUE_GROUP
Lists queues that participate in an absolute priority scheduling (APS) queue group.

If both the **FAIRSHARE** and **APS_PRIORITY** parameters are enabled in the same queue, the FAIRSHARE_QUEUES are not displayed. These queues are instead displayed as QUEUE_GROUP.

DISPATCH_ORDER
The **DISPATCH_ORDER=QUEUE** parameter is set in the master queue. Jobs from this queue are dispatched according to the order of queue priorities first, then user fairshare priority. Within the queue, dispatch order is based on user share quota. Share quotas avoid job dispatch from low-priority queues for users with higher fairshare priority.

USER_SHARES
A list of [*user_name, share*] pairs. The *user_name* is either a user name or a user group name. The *share* is the number of shares of resources that are assigned to the user or user group. A consumer receives a portion of the resources proportional to that consumer's share that is divided by the sum of the shares of all consumers that are specified in the queue.

DEFAULT HOST SPECIFICATION
The default host or host model that is used to normalize the CPU time limit of all jobs.

Use the **lsinfo** command to view a list of the CPU factors that are defined for the hosts in your cluster. The CPU factors are configured in the `lsf.shared` file.

The appropriate CPU scaling factor of the host or host model is used to adjust the actual CPU time limit at the execution host (the **CPULIMIT** parameter in the `lsb.queues` file). The **DEFAULT_HOST_SPEC** parameter in `lsb.queues` overrides the system **DEFAULT_HOST_SPEC** parameter in the `lsb.params` file. If you explicitly give a host specification when you submit a job with the **bsub -c** *cpu_limit*[*/host_name | /host_model*] command, the job-level specification overrides the values that are defined in the `lsb.params` and `lsb.queues` files.

RUN_WINDOWS
The time windows in a week during which jobs in the queue can run.

When a queue is out of its window or windows, no job in this queue is dispatched. In addition, when the end of a run window is reached, any running jobs from this queue are suspended until the beginning of the next run window, when they are resumed. The default is no restriction, or always open.

DISPATCH_WINDOWS
Dispatch windows are the time windows in a week during which jobs in the queue can be dispatched.

When a queue is out of its dispatch window or windows, no job in this queue is dispatched. Jobs that are already dispatched are not affected by the dispatch windows. The default is no restriction, or always open (that is, twenty-four hours a day, seven days a week). Dispatch windows are only applicable to batch jobs. Interactive jobs that are scheduled by LIM are controlled by another set of dispatch windows. Similar dispatch windows can be configured for individual hosts.

A window is displayed in the format *begin_time*-*end_time*. Time is specified in the format [*day*:]*hour*[:*minute*], where all fields are numbers in their respective legal ranges: 0(Sunday)-6 for *day*, 0-23 for *hour*, and 0-59 for *minute*. The default value for *minute* is 0 (on the hour). The default value for *day* is every day of the week. The *begin_time* and *end_time* of a window are separated by a dash (-), with no blank characters (SPACE and TAB) in between. Both *begin_time* and *end_time* must be present for a window. Windows are separated by blank characters.

USERS
A list of users who are allowed to submit jobs to this queue. LSF administrators can submit jobs to the queue even if they are not listed here.

User group names have a slash (/) added at the end of the group name. Use the **bugroup** command to see information about user groups.

If the fairshare scheduling policy is enabled, users and LSF administrators cannot submit jobs to the queue unless they also have a share assignment.

HOSTS
A list of hosts where jobs in the queue can be dispatched.

Host group names have a slash (/) added at the end of the group name. Use the **bmgroup** command to see information about host groups.

NQS DESTINATION QUEUES
A list of NQS destination queues to which this queue can dispatch jobs.

When you submit a job with the **bsub -q** *queue_name* command, and the specified queue is configured to forward jobs to the NQS system, LSF routes your job to one of the NQS destination queues. The job runs on an NQS batch server host, which is not a member of the LSF cluster. Although the job runs on an NQS system outside the LSF cluster, it is still managed by LSF in almost the same way as jobs that run inside the cluster. Your batch jobs might be transparently sent to an NQS system to run. You can use any supported user interface, including LSF commands and NQS commands (see the **lsnqs** command) to submit, monitor, signal, and delete your batch jobs that are running in an NQS system.

ADMINISTRATORS
A list of queue administrators. The users whose names are specified here are allowed to operate on the jobs in the queue and on the queue itself.

PRE_EXEC
The job-based pre-execution command for the queue. The **PRE_EXEC** command runs on the execution host before the job that is associated with the queue is dispatched to the execution host (or to the first host selected for a parallel batch job).

POST_EXEC
The job-based post-execution command for the queue. The **POST_EXEC** command runs on the execution host after the job finishes.

HOST_PRE_EXEC
The host-based pre-execution command for the queue. The **HOST_PRE_EXEC** command runs on all execution hosts before the job that is associated with the queue is dispatched to the execution hosts. If a job-based pre-execution **PRE_EXEC** command is defined at the queue-level, application-level, or job-level, the **HOST_PRE_EXEC** command runs before **PRE_EXEC** command of any level. The host-based pre-execution command cannot be run on Windows systems.

HOST_POST_EXEC
The host-based post-execution command for the queue. The **HOST_POST_EXEC** command runs on all execution hosts after the job finishes. If a job-based post-execution **POST_EXEC** command is defined at the queue-level, application-level, or job-level, the **HOST_POST_EXEC** command runs after **POST_EXEC** command of any level. The host-based post-execution command cannot be run on Windows systems.

LOCAL_MAX_PREEXEC_RETRY_ACTION
The action to take on a job when the number of times to attempt its pre-execution command on the local cluster (**LOCAL_MAX_PREEXEC_RETRY** value) is reached.

REQUEUE_EXIT_VALUES
Jobs that exit with these values are automatically requeued.

RES_REQ
Resource requirements of the queue. Only the hosts that satisfy these resource requirements can be used by the queue.

RESRSV_LIMIT
Resource requirement limits of the queue. Queue-level **RES_REQ** `rusage` values (set in the `lsb.queues` file) must be in the range set by RESRSV_LIMIT, or the queue-level **RES_REQ** value is ignored. Merged **RES_REQ** `rusage` values from the job and application levels must be in the range that is shown by the RESRSV_LIMIT, or the job is rejected.

Maximum slot reservation time
The maximum time in seconds a slot is reserved for a pending job in the queue. For more information, see the **SLOT_RESERVE=MAX_RESERVE_TIME[n]** parameter in the `lsb.queues` file.

RESUME_COND
The conditions that must be satisfied to resume a suspended job on a host.

STOP_COND
The conditions that determine whether a job that is running on a host needs to be suspended.

JOB_STARTER
An executable file that runs immediately before the batch job, taking the batch job file as an input argument. All jobs that are submitted to the queue are run through the job starter, which is used to create a specific execution environment before the jobs themselves are processed.

CHUNK_JOB_SIZE
Chunk jobs only. Specifies the maximum number of jobs that are allowed to be dispatched together in a chunk job. All of the jobs in the chunk are scheduled and dispatched as a unit rather than individually. The ideal candidates for job chunking are jobs that typically take 1 - 2 minutes to run.

SEND_JOBS_TO
LSF multicluster capability. List of remote queue names to which the queue forwards jobs.

RECEIVE_JOBS_FROM
LSF multicluster capability. List of remote cluster names from which the queue receives jobs.

PREEMPTION

PREEMPTIVE
The queue is preemptive. Jobs in this queue can preempt running jobs from lower-priority queues, even if the lower-priority queues are not specified as preemptive.

PREEMPTABLE
The queue is preemptable. Running jobs in this queue can be preempted by jobs in higher-priority queues, even if the higher-priority queues are not specified as preemptive.

RC_ACCOUNT
The account name (tag) that is assigned to hosts borrowed through LSF resource connector, so that they cannot be used by other user groups, users, or jobs.

RC_HOSTS
The list of Boolean resources that represent the host resources that LSF resource connector can borrow from a resource provider.

RERUNNABLE
If the RERUNNABLE field displays yes, jobs in the queue are rerunnable. Jobs in the queue are automatically restarted or rerun if the execution host becomes unavailable. However, a job in the queue is not restarted if you remove the rerunnable option from the job.

CHECKPOINT
If the CHKPNTDIR field is displayed, jobs in the queue are checkpointable. Jobs use the default checkpoint directory and period unless you specify other values. A job in the queue is not checkpointed if you remove the checkpoint option from the job.

CHKPNTDIR
Specifies the checkpoint directory by using an absolute or relative path name.

CHKPNTPERIOD
Specifies the checkpoint period in seconds.

Although the output of the **bqueues** command reports the checkpoint period in seconds, the checkpoint period is defined in minutes. The checkpoint period is defined with the **bsub -k** "*checkpoint_dir* []" option, or in the lsb.queues file.

JOB CONTROLS
The configured actions for job control. See the **JOB_CONTROLS** parameter in the lsb.queues file.

The configured actions are displayed in the format [*action_type*, *command*] where *action_type* is either SUSPEND, RESUME, or TERMINATE.

ADMIN ACTION COMMENT
If the LSF administrator specified an administrator comment with the -C option of a queue control commands (**qclose**, **qopen**, **qact**, **qinact**, or **qhist**), the comment text is displayed.

SLOT_SHARE
Share of job slots for queue-based fairshare. Represents the percentage of running jobs (job slots) in use from the queue. The SLOT_SHARE value must be greater than zero.

The sum of SLOT_SHARE for all queues in the pool does not need to be 100%. It can be more or less, depending on your needs.

SLOT_POOL
Name of the pool of job slots the queue belongs to for queue-based fairshare. A queue can belong to only one pool. All queues in the pool must share hosts.

MAX_SLOTS_IN_POOL
Maximum number of job slots available in the slot pool the queue belongs to for queue-based fairshare. Defined in the first queue of the slot pool.

USE_PRIORITY_IN_POOL
Queue-based fairshare only. After job scheduling occurs for each queue, this parameter enables LSF to dispatch jobs to any remaining slots in the pool in first-come first-served order across queues.

NO_PREEMPT_INTERVAL
The uninterrupted running time (minutes) that must pass before preemption is permitted. Configured in the lsb.queues file.

MAX_TOTAL_TIME_PREEMPT
The maximum total preemption time (minutes) above which preemption is not permitted. Configured in the lsb.queues file.

SHARE_INFO_FOR
User shares and dynamic priority information based on the scheduling policy in place for the queue.

USER/GROUP
Name of users or user groups who have access to the queue.

SHARES
Number of shares of resources that are assigned to each user or user group in this queue, as configured in the file lsb.queues. The shares affect dynamic user priority for when fairshare scheduling is configured at the queue level.

PRIORITY
Dynamic user priority for the user or user group. Larger values represent higher priorities. Jobs belonging to the user or user group with the highest priority are considered first for dispatch.

In general, users or user groups with the following properties have higher PRIORITY:

- Larger SHARES
- Fewer STARTED and RESERVED jobs
- Lower CPU_TIME and RUN_TIME

STARTED
Number of job slots that are used by running or suspended jobs that are owned by users or user groups in the queue.

RESERVED
Number of job slots that are reserved by the jobs that are owned by users or user groups in the queue.

CPU_TIME
Cumulative CPU time that is used by jobs that are run from the queue. Measured in seconds, to one decimal place.

LSF calculates the cumulative CPU time by using the actual (not normalized) CPU time. LSF uses a decay factor such that 1 hour of recently used CPU time decays to 0.1 hours after an interval of time that is specified by the **HIST_HOURS** parameter in the `lsb.params` file. The default for the **HIST_HOURS** parameter is 5 hours.

RUN_TIME
Wall-clock run time plus historical run time of jobs of users or user groups that are run in the queue. Measured in seconds.

LSF calculates the historical run time by using the actual run time of finished jobs. LSF uses a decay factor such that 1 hour of recently used run time decays to 0.1 hours after an interval of time that is specified by the **HIST_HOURS** parameter in the `lsb.params` file. The default for the **HIST_HOURS** parameter is 5 hours. Wall-clock run time is the run time of running jobs.

ADJUST
Dynamic priority calculation adjustment that is made by the user-defined fairshare plug-in(`libfairshareadjust.*`).

The fairshare adjustment is enabled and weighted by the parameter **FAIRSHARE_ADJUSTMENT_FACTOR** in the `lsb.params` file.

RUN_TIME_FACTOR
The weighting parameter for run_time within the dynamic priority calculation. If not defined for the queue, the cluster-wide value that is defined in the `lsb.params` file is used.

CPU_TIME_FACTOR
The dynamic priority calculation weighting parameter for CPU time. If not defined for the queue, the cluster-wide value that is defined in the `lsb.params` file is used.

ENABLE_HIST_RUN_TIME
Enables the use of historic run time (run time for completed jobs) in the dynamic priority calculation. If not defined for the queue, the cluster-wide value that is defined in the `lsb.params` file is used.

RUN_TIME_DECAY
Enables the decay of run time in the dynamic priority calculation. The decay rate is set by the parameter **HIST_HOURS** (set for the queue in the `lsb.queues` file or set for the cluster in the `lsb.params` file). If not defined for the queue, the cluster-wide value that is defined in the `lsb.params` file is used.

HIST_HOURS
Decay parameter for CPU time, run time, and historic run time. If not defined for the queue, the cluster-wide value that is defined in the `lsb.params` file is used.

FAIRSHARE_ADJUSTMENT_FACTOR
Enables and weights the dynamic priority calculation adjustment that is made by the user-defined fairshare plug-in(`libfairshareadjust.*`). If not defined for the queue, the cluster-wide value that is defined in the `lsb.params` file is used.

RUN_JOB_FACTOR
The dynamic priority calculation weighting parameter for the number of job slots that are reserved and in use by a user. If not defined for the queue, the cluster-wide value that is defined in the `lsb.params` file is used.

COMMITTED_RUN_TIME_FACTOR
The dynamic priority calculation weighting parameter for committed run time. If not defined for the queue, the cluster-wide value that is defined in the `lsb.params` file is used.

JOB_SIZE_LIST
A list of job sizes (number of tasks) allowed on this queue, including the default job size that is assigned if the job submission does not request a job size. Configured in the `lsb.queues` file.

PEND_TIME_LIMIT
The pending time limit for a job in the queue. If a job remains pending for longer than this specified time limit, LSF sends a notification to IBM Spectrum LSF RTM. Configured in the `lsb.queues` file.

ELIGIBLE_PEND_TIME_LIMIT
The eligible pending time limit for a job in the queue. If a job remains in an eligible pending state for longer than this specified time limit, LSF sends a notification to IBM Spectrum LSF RTM. Configured in the `lsb.queues` file.

RELAX_JOB_DISPATCH_ORDER
If the **RELAX_JOB_DISPATCH_ORDER** parameter is configured in the `lsb.params` or `lsb.queues` file, the allocation reuse duration, in minutes, is displayed.

NORM_FS
Normalized fairshare factors, if the factors are not zero.

Recursive Share Tree Output (-r)

In addition to the fields displayed for the `-l` option, the `-r` option displays the following fields:

SCHEDULING POLICIES

FAIRSHARE
The **bqueues -r** command recursively displays the entire share information tree that is associated with the queue.

See also

bugroup, **nice**, **getrlimit**, `lsb.queues`, **bsub**, **bjobs**, **bhosts**, **badmin**, **mbatchd**

Chapter 41. bread

Reads messages and attached data files from a job.

Synopsis

`bread` [`-i` *message_index*] [`-a` *file_name*] [`-N`] [`-w`]*job_ID* | "*job_ID*[*index*]" | `-J` *job_name*

`bread` [`-h` | `-V`]

Description

Reads messages and data posted to an unfinished job with the **bpost** command.

By default, displays the message description text of the job. By default, operates on the message with index `0`.

You can read messages and data from a job until it is cleaned from the system. You cannot read messages and data from done or exited jobs.

If you specify a job ID, you can take the following actions:

- You can get read messages of jobs submitted by other users, but you cannot read data files that are attached to jobs submitted by other users.
- You can read only data files that are attached to your own jobs.
- Root and LSF administrators can read messages of jobs that are submitted by other users.
- Root and LSF administrators cannot read data files that are attached to jobs submitted by other users.

Job names are not unique; if you specify the `-J` *job_name* option the following actions are supported:

- You can read messages and data only from your own jobs.
- You cannot read messages and data from jobs that are submitted by other users.
- Root and the LSF administrators can read messages and data only from their own jobs.

The **bstatus** command is equivalent to the following command:

```
bread -i 0
```

Options

-a *file_name*
Gets the text message and copies the data file attached to the specified message index of the job to the file specified by *file_name*. Data files cannot be attached to LSF multicluster capability jobs.

If you do not specify a message index, copies the attachment of message index `0` to the file. The job must have an attachment, and you must specify a name for the file you are copying the attachment to. If the file exists, the `-a` option overwrites it with the new file.

By default, the `-a` option gets the attachment file from the directory that is specified by the **JOB_ATTA_DIR** parameter. If the **JOB_ATTA_DIR** parameter is not specified, job message attachments are saved in the `LSB_SHAREDIR/info/` directory.

-i *message_index*
Specifies the message index to be retrieved.

The default *message_index* is `0`.

-N
Displays the `NOTIFICATION` field to indicate whether the message was sent to LSF Application Center Notifications. A hyphen (`-`) indicates that this was not sent to LSF Application Center

Notifications (either because it is a normal message that was sent without the `-N` option or the **LSF_AC_PNC_URL** parameter is not configured correctly in the `lsf.conf` file).

-w
Wide format. Displays information without truncating fields.

job_ID* | "*job_ID[index]*" | -J *job_name
Required. Specify the job to operate on.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Example

```
bpost -i 1 -N INFO -d "step 1" -a step1.out 2500

bread -i 1 -N -a step2.in 2500
JOBID      MSG_ID FROM      POST_TIME      DESCRIPTION  NOTIFICATION
2500       1      user1     May 19 13:59   step 1       SENT
```

Displays the message description text `step 1` for message index 1 of job 2500, displays whether the message is sent to LSF Application Center Notifications, and copies the data in the file `step1.out` attached to message 1 to the file `step2.in`.

See also

bpost, **bstatus**, **bsub**, **JOB_ATTA_DIRJOB_ATTA_DIR**

Chapter 42. brequeue

Kills and requeues a job.

Synopsis

`brequeue [-a] [-d] [-e] [-H] [-p] [-r] [-J job_name | -J "job_name[index_list]"] [-u user_name | -u all] [job_ID | "job_ID[index_list]" ...]`

`brequeue [-h | -V]`

Description

You can use the **brequeue** command only on jobs that you own. root or the LSF cluster administrator or LSF group administrator can requeue jobs for any user.

Kills a running (RUN), user-suspended (USUSP), or system-suspended (SSUSP) job and returns it to the queue. A job that is killed and requeued retains its submit time but is dispatched according to its requeue time. When the job is requeued, it is assigned the PEND status or PSUSP if the `-H` option is used. after the job is dispatched, it starts over from the beginning. The requeued job keeps the same job ID.

When the **JOB_INCLUDE_POSTPROC=Y** parameter is set in the `lsb.params` file or in an application profile in the `lsb.applications` file, job requeue occurs only after post-execution processing, not when the job finishes. When used for host-based post-execution processing, configure a longer time period to allow the operation to run.

Use the **brequeue** command to requeue job arrays or job array elements.

By default, the **brequeue** command kills and requeues your most recently submitted job when no job ID is specified.

In the LSF multicluster capability lease model, you can use the **brequeue** command only on jobs in local queues. A job that is killed and requeued is assigned a new job ID on the execution cluster, but it retains the same job ID on the submission cluster. For example, a job that is submitted from cluster A that is killed and requeued and then runs on execution cluster B is assigned a new job ID on cluster B. However, when the **bjobs** command runs from submission cluster A, the job is displayed with the original job ID. When the **bjobs** command runs from execution cluster B, the job is displayed with the new job ID.

In the LSF multicluster capability job forwarding model, use the **brequeue -p** command to requeue specified remote pending jobs. Use the **brequeue -a** command to requeue all non-pending jobs (including running jobs, suspended jobs, jobs with EXIT or DONE status) in the local cluster. The **brequeue -a** command does not requeue pending jobs in the local cluster.

The only difference between the `-p` and `-a` options is the job dispatch order. Running the **brequeue -p** command on the submission cluster requeues a remote job to the top of the queue so that the requeued job is dispatched first no matter which position it is in the pending job list. The **brequeue -a** command sends the remote job to the end of the queue the same way as in the local cluster.

When absolute job priority scheduling (APS) is configured in the queue, specified requeued jobs are treated as newly submitted jobs for APS calculation. The job priority, system, and the ADMIN APS factors are reset on requeue.

When you use multi-phase rusage resource requirement strings, such as with the **bsub -R** command, the requeued job is treated as a new job and resources are reserved from the beginning of the first phase.

Options

-a

Requeues all local non-pending jobs, including running jobs, suspending jobs, jobs with EXIT or DONE status, and pending remote jobs. It does not requeue pending jobs in the local cluster.

-d
Requeues jobs that finished running with DONE job status.

-e
Requeues jobs that terminated abnormally with EXIT job status.

-H
Requeues jobs to PSUSP job status.

-p
In the LSF multicluster capability job forwarding model, requeues specified jobs that are pending in a remote cluster for LSF multicluster capability job forwarding modes.

-r
Requeues jobs that are running.

-J *job_name* | -J "*job_name[index_list]*"
Operates on the specified job.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-u *user_name* | -u all
Operates on the specified user's jobs or all jobs. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) at a Windows command prompt. On a UNIX or Linux command line, use a double backslash (*DOMAIN_NAME**user_name*).

Only `root` and LSF administrators can requeue jobs that are submitted by other users.

***job_ID* |"*job_ID[index_list]*"**
Operates on the specified job or job array elements.

The value of 0 for *job_ID* is ignored.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Limitations

The **brequeue** command cannot be used on interactive batch jobs. The **brequeue** command kills only interactive batch jobs, it does not restart them.

Chapter 43. bresize

Decreases or increases tasks that are allocated to a running resizable job, or cancels pending job resize allocation requests.

Synopsis

`bresize` *subcommand*

`bresize` [`-h` | `-V`]

Subcommand List

`release` [`-c`] [`-rnc` *resize_notification_cmd* | `-rncn`] *released_host_specification job_ID*

`request` [`-c`] [`-rnc` *resize_notification_cmd* | `-rncn`] [*min_task*`,`] *tasks job_ID*

`cancel` *job_ID*

Description

Use the **bresize release** command to explicitly release tasks from a running job. When you release tasks from an allocation, a minimum of one task on the first execution host must be retained. Only hosts (and not host groups or compute units) can be released by using the **bresize release** command. When you release tasks from compound resource requirements, you can release only tasks that are represented by the last term of the compound resource requirement. To release tasks in earlier terms, run **bresize release** repeatedly to release tasks in subsequent last terms.

Use the **bresize request** command to trigger a manual request for additional allocated tasks. LSF pends the request if the queue cannot meet the minimum tasks request or if the request is over the **TASKLIMIT** value for the queue or application profile. Changing the **TASKLIMIT** value does not affect any requests that are already accepted. For compound resource requirements, the request only applies to the last term. For alternative resource requirements, the request only applies to the term that was used for the initial task allocation. For autoresizable jobs, if there is pending demand, you must first cancel the previous pending demand by running the **brequest request -c** or **bresize cancel** commands. After triggering this manual request, the job is no longer autoresizable unless you requeue or rerun the job.

Use **bresize cancel** to cancel a pending allocation request for the specified job ID. The active pending allocation request is generated by LSF automatically for autoresizable jobs. If the job does not have an active pending request, the command fails with an error message.

By default, only cluster administrators, queue administrators, root, and the job owner are allowed to run **bresize** to change job allocations.

User group administrators are allowed to run **bresize** to change the allocation of jobs within their user groups.

Options

-c
: Optional. Cancel the active pending resource request when you release tasks from existing allocation (for the **release** subcommand) or before accepting new grow requests (for the **request** subcommand). By default, the command releases tasks only for jobs without pending requests.

-rnc *resize_notification_cmd*
: Optional. Specify the name of an executable file to be started on the first execution host when the job allocation was modified. This setting applies only to this request, which overrides any notification command that is specified in the **bsub** command or an application profile. The resize notification command runs under the user account of the job owner.

-rncn
Cancels the resize notification command at both job-level and application-level. This setting applies only to this request.

released_host_specification
Required with the **release** subcommand. Defines the list of hosts to be released. The following syntax is the EBNF definition of the released host specification:

```
<released_host_spec> ::= all | all <exclude_host_list_spec>
 | <host_list_spec><host_list_spec> ::= <host_spec>
 | <host_list_spec><host_spec><exclude_host_list_spec>
 ::= <exclude_host_spec> | <exclude_host_list_spec> <exclude_host_spec> <exclude_host_spec>
 ::= ~<host_spec> <host_spec>
 ::= [<positive_integer>*]<host_name>
```

all
Specifies all the tasks currently being used by the job. If the `all` option is used alone, it means release every task except one task from the first execution node. The `all` option can also be used with a list of hosts to exclude with the tilde (not) operator (~).

host_spec
Release the number of tasks that are specified by *positive_integer* on the host that is specified by *host_name*. If the number of tasks is not specified, all tasks on the specified host are released.

~
Specifies hosts to exclude when you release tasks. Tasks on the specified hosts are not released. The tilde (not) operator (~) must be used together with `all` keyword.

min_task
Optional. The minimum number of tasks to be added to the job. LSF pends the manual job resize request if the queue cannot meet the minimum tasks request.

task
Required for the **request** subcommand. The maximum number of tasks to be added to the job.

job_ID
Required. The job ID of the job to be resized. For job arrays, this also includes the array index.

-h
Prints command usage to `stderr` and exits.

-V
Prints release version to `stderr` and exits.

Examples

The following examples all resize a job with ID 100.

For a job that uses 8 tasks across 4 nodes (2 on `hostA`, 2 on hostB, 2 on `hostC`, and 2 on `hostD`, the following command releases all tasks except the tasks on `hostA`. After the tasks are released, the job allocation becomes just 2 on `hostA`:

```
bresize release "all ~hostA" 100
```

The following command releases all tasks except one task from `hostA`. After the tasks are released, the job allocation becomes 1 on `hostA`:

```
bresize release all 100 or bresize release "all ~1*hostA" 100
```

The following command releases 1 task from each of four hosts. After the tasks are released, the job allocation becomes 1 on `hostA`, 1 on `hostB`, 1 on `hostC`, and 1 on `hostD`:

```
bresize release "1*hostA 1*hostB 1*hostC 1*hostD"  100
```

The following command requests a total of 10 tasks to be added to the job. The request pends if LSF cannot add all 10 requested tasks to the job:

```
bresize request 10 100
```

The following command requests a minimum of 4 and a total of 10 tasks to be added to the job. The request pends if LSF cannot add at least 4 tasks to the job.

```
bresize request 4,10 100
```

See also

bsub, `lsb.applications`

Chapter 44. bresources

Displays information about resource reservation, resource limits, and guaranteed resource policies.

Synopsis

`bresources -s` [*resource_name* ...]

`bresources -g` [`-l` [`-q` *queue_name*] [`-m`]] [guaranteed_resource_pool_name ...]

`bresources` [`-h` | `-V`]

`bresources -p`

Description

By default, the **bresources** command displays all resource limit configurations in the `lsb.resources` file. The default **bresources** command output is the same as the **blimits -c** command.

Options

-s
Displays per-resource reservation configurations from the **ReservationUsage** section of `lsb.resources`.

resource_name ...
Used with `-s`, displays reservation configurations about the specified resource.

-g
Displays the configuration and status of the guaranteed resource pool that is configured in the **GuaranteedResourcePool** section of `lsb.resources`. The following information about each guaranteed resource pool is displayed: name, type, status, available resources, unused resources, total configured resource guarantees, and number of guaranteed resources currently unused.

-l
With `-g`, displays configuration and status information of the guaranteed resource pool that is configured in the **GuaranteedResourcePool** section of `lsb.resources` in a long multiline format. The `-l` option displays the following information:

- Description
- Distribution of guarantees among service classes
- Special policies
- Configured hosts list
- Static resource requirement `select` string
- Administrator
- The following information for each guarantee that is made from the resource pool:
  - Name
  - Resources guaranteed
  - Resources in use

-m
With `-g` and `-l`, displays the names of all hosts included in each guaranteed resource pool configuration from the `GuaranteedResourcePool` section of the `lsb.resources` file.

-q
Helps you understand how the guarantee policy works when combined with queue-based preemption. Administrators can determine the number of guarantee resources available through preemption to a preemptive queue. The `-q` option takes effect only for package pools. When the `-q`

option is specified with a name of a preemptive queue, the values that are displayed are shown about the specified queue.

resource_pool ...
Displays information about the specified resource pool.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

-p
Displays the currently defined energy aware scheduling policies and exits. Shows the `PowerPolicy` settings as they are in the `lsb.resources` file. An extra line is included with the `PowerPolicy` settings to indicate whether the policy is applied (`Y`) or not (`N`).

Guaranteed resource pool output (-g)

POOL_NAME
Configured name of the guaranteed resource pool.

TYPE
Configured type of guaranteed resource pool. The pool can include the following resources:

- `hosts`
- `slots`
- `packages`, where each unit guaranteed is composed of a number of slots, and some amount of memory together on the same host.
- `resources` that are managed by LSF License Scheduler.

STATUS
The following values are displayed:

- ok
- `unknown`
- `overcommitted`, where total resources in the pool are less than the number guaranteed. The guarantee commitments cannot all be met concurrently.
- `close_loans`, where lending is suspended due to pending demand.

  This state occurs only when the **CLOSE_ON_DEMAND** [parameter is set in the `LOAN_POLICIES` section, and at least one job with a guarantee in the pool is not using all of its configured guarantee.

TOTAL
Number of resources that are included in the guaranteed resource pool.

FREE
Number of unused resources within the guaranteed resource pool.

GUARANTEE CONFIGURED
Configured number of guaranteed resources.

GUARANTEE USED
Number of guarantees that are used.

Long output (-gl)

In addition to the fields included in the guaranteed resource pool output (`-g` option), the long output includes the following fields.

GUARANTEED RESOURCE POOL
Name and description of guaranteed resource pool.

DISTRIBUTION
Configured distribution of guarantees among service classes.

LOAN_POLICIES
Configured policies.

HOSTS
Configured hosts list.

OTHER USE
The amount of the resources that are used by other jobs that are running on shared hosts and are not owner or loaning jobs.

In the output table that is organized by owner, this field is the amount of the resources that are used by jobs of the service class on shared hosts in the guarantee shared pool (that is, hosts that are marked as "-"). This usage does not count towards the owner's guarantee limits.

In the output table that is organized by host, this field is the amount of the resources that are used on the host by other jobs that are not owner or loaning jobs.

This field is hidden if the **SIMPLIFIED_GUARANTEE** parameter in the `lsb.params` file is disabled (that is, set to N or n) or is not defined.

CONSUMERS
Service classes with guarantees in the pool.

GUARANTEE CONFIGURED
Number of resources in the pool that are guaranteed to the service class.

GUARANTEE USED
Number of resources currently in use by the service class to meet the guarantee. After the guarantee is met, other jobs from the service class that run in the resource pool do not count towards the guarantee, and are not included. Resource use includes both running and suspended jobs.

TOTAL USED
Total number of resources that are used in the pool by the service class. Resource use includes both running and suspended jobs.

ADMINISTRATORS
Configured administrators that can manage the corresponding guaranteed resource pool.

Long output with hosts (-glm)

In addition to fields included in the long output (option `-gl`), hosts currently in the resource pool are listed.

Examples

```
bresources -g
                                                    GUARANTEE GUARANTEE
POOL_NAME         TYPE     STATUS    TOTAL    FREE   CONFIGURED      USED
slotPool         slots         ok        4       4            0         0
```

```
bresources -g -l
GUARANTEED RESOURCE POOL: slotPool
guaranteed slot policy

TYPE: slots
DISTRIBUTION: [sla1, 0%]
LOAN_POLICIES: QUEUES[normal] DURATION[10]
HOSTS: HostA
STATUS: ok
ADMINISTRATORS: userA


RESOURCE SUMMARY:
   TOTAL                          4
   FREE                           4
   GUARANTEE CONFIGURED           0
   GUARANTEE USED                 0

                                    GUARANTEE  GUARANTEE     TOTAL
```

```
CONSUMERS                              CONFIGURED      USED      USED
sla1                                            0         0         0
```

```
bresources -g -l -m
GUARANTEED RESOURCE POOL: slotPool
guaranteed slot policy

TYPE: slots
DISTRIBUTION: [sla1, 0%]
LOAN_POLICIES: QUEUES[normal] DURATION[10]
HOSTS: HostA
STATUS: ok
ADMINISTRATORS: userA

RESOURCE SUMMARY:
  TOTAL                            4
  FREE                             4
  GUARANTEE CONFIGURED             0
  GUARANTEE USED                   0

                                    GUARANTEE  GUARANTEE      TOTAL
CONSUMERS                          CONFIGURED       USED       USED
sla1                                        0          0          0

Hosts currently in the resource pool:
HostA
```

```
bresources -p
    Begin PowerPolicy
            NAME = policy_night
            HOSTS = hostGroup1 host3
            TIME_WINDOW= 23:00-8:00
            APPLIED = No
       End PowerPolicy
```

Chapter 45. brestart

Restarts checkpointed jobs.

Synopsis

`brestart` [*bsub_options*] [`-f`] *checkpoint_dir* [*job_ID* | "*job_ID*[*index*]"]

`brestart` [`-h` | `-V`]

Option List

`-B`

`-f`

`-N` | `-Ne`

`-x`

`-a` "*esub_application*[([*argument*[,*argument*...])]..."

`-b` *begin_time*

`-C` *core_limit*

`-c` [*hour*:]*minute*[/*host_name* | /*host_model*]

`-D` *data_limit*

`-E` "*pre_exec_command* [*argument* ...]"

`-F` *file_limit*

`-m` "*host_name*[+[*pref_level*]] | *host_group*[+[*pref_level*]] ..."

`-G` *user_group*

`-M` *mem_limit*

`-q` "*queue_name* ..."

`-R` "*res_req*" [`-R` "*res_req*" ...]

`-S` *stack_limit*

`-t` *term_time*

`-w` '*dependency_expression*' [`-ti`]

`-W` *run_limit*[/*host_name*| /*host_model*]

checkpoint_dir [*job_ID* | "*job_ID*[*index*]"]

Description

Restarts a checkpointed job by using the checkpoint files that are saved in the directory *checkpoint_dir/last_job_ID/*. Only jobs that are successfully checkpointed can be restarted.

Jobs are resubmitted and assigned a new job ID. The checkpoint directory is renamed by using the new job ID, *checkpoint_dir/new_job_ID/*.

The file path of the checkpoint directory can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

By default, jobs are restarted with the same output file and file transfer specifications, job name, window signal value, checkpoint directory and period, and rerun options as the original job.

A job can be restarted on another host under the following conditions for both hosts:

- Must be binary compatible
- Must run the same OS version
- Have access to the executable files
- Have access to all open files (LSF must locate them with an absolute path name)
- Have access to the checkpoint directory

The environment variable **LSB_RESTART** is set to Y when a job is restarted.

LSF invokes the **erestart** executable file in the LSF_SERVERDIR directory to restart the job.

Only the **bsub** options that are listed here can be used with the **brestart** command.

Like the **bsub**command, the **brestart** command calls the master **esub** file (the **mesub** file), which calls the executable file named **esub** (without *.application*) if it exists in the LSF_SERVERDIR directory. The **mesub** file also calls any mandatory **esub** executable files that are configured by an LSF administrator. Only **esub** executable files that are called by the **bsub** command can change the job environment on the submission host. An **esub** file that is called by the **brestart** command cannot change the job environment. Arguments for the **esub** executable files can also be modified.

You can use the **brestart -R** command to specify new resource requirements when you restart a checkpointable job. The new resource requirements must be mem or swap. You can use the **brestart** to specify multiple -R options for multiple resource requirement strings, specify compound resource requirements, and specify alternative resource requirements.

Options

The following option applies only to the **brestart** command.

-f
Forces the job to be restarted even if non-restartable conditions exist (these conditions are operating system specific).

See also

bsub, **bjobs**, **bmod**, **bqueues**, **bhosts**, **bchkpnt**, **lsbqueues**, **echkpnt**, **erestart**, **mbatchd**

Chapter 46. bresume

Resumes one or more suspended jobs.

Synopsis

`bresume` [`-app` *application_profile_name*] [`-g` *job_group_name*] [`-J` *job_name*] [`-m` *host_name*] [`-q` *queue_name*] [`-sla` *service_class_name*] [`-u` *user_name* | `-u` *user_group* | `-u all`] [`0`]

`bresume` [*job_ID* | "*job_ID*[*index_list*]"] ...

`bresume` [`-h` | `-V`]

Description

Sends the `SIGCONT` signal to resume one or more of your suspended jobs.

Only root and LSF administrators can operate on jobs that are submitted by other users. You cannot resume a job that is not suspended. Using the **bresume** command on a job that is not in either the PSUSP or the USUSP state has no effect.

You must specify a job ID or the `-g`, `-J`, `-m`, `-u`, or `-q` option. Specify `0` (zero) to resume multiple jobs.

You can also use the **bkill -s CONT** command to send the resume signal to a job.

If a signal request fails to reach the job execution host, LSF retries the operation later when the host becomes reachable. LSF retries the most recent signal request.

Jobs that are suspended by the administrator can be resumed only by the LSF administrator or root. Users do not have permission to resume a job that is suspended by another user or the administrator. Administrators or root can resume jobs that are suspended by users or administrators.

ENABLE_USER_RESUME parameter in the lsb.params file

If the `ENABLE_USER_RESUME=Y` parameter is set in the `lsb.params` file, users can resume their own jobs that are suspended by the administrator.

Options

0
Resumes all the jobs that satisfy other options (`-g`, `-m`, `-q`, `-u`, and `-J`).

-app *application_profile_name*
Resumes only jobs that are associated with the specified application profile. You must specify an existing application profile.

-g *job_group_name*
Resumes only jobs in the specified job group.

-J *job_name*
Resumes only jobs with the specified name.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (`*`) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-m *host_name*
Resumes only jobs that are dispatched to the specified host.

-q *queue_name*
Resumes only jobs in the specified queue.

-sla *service_class_name*
Resume jobs that belong to the specified service class.

Use the **bsla** command to display the properties of service classes that are configured in the `lsb.serviceclasses` file and dynamic information about the state of each configured service class.

-u *user_name* | -u *user_group* | -u all
Resumes only jobs that are owned by the specified user or group, or all users if the reserved user name `all` is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) on a Windows command prompt or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX or Linux command line.

***job_ID* ... | "*job_ID*[*index_list*]" ...**
Resumes only the specified jobs. Jobs that are submitted by any user can be specified here without using the `-u` option.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Examples

```
bresume -q night 0
```

Resumes all suspended jobs that belong to the user in the `night` queue. If the user is the LSF administrator, resumes all suspended jobs in the `night` queue.

```
bresume -g /risk_group 0
```

Resumes all suspended jobs in the job group `/risk_group`.

See also

bsub, **bjobs**, **bqueues**, **bhosts**, **bstop**, **bkill**, **bgadd**, **bgdel**, **bjgroup**, **bparams**, **bapp**, **mbatchd**, **kill**, **signal**, `lsb.params`, `lsb.applications`

Chapter 47. brlainfo

Displays host topology information.

Synopsis

`brlainfo` [`-l`] [*host_name* ...]

`brlainfo` [`-h` | `-V`]

Description

The **`brlainfo`** command contacts the LSF topology adapter (RLA) on the specified host and presents topology information to the user. By default, displays information about all hosts that run the RLA.

Options

-l
Long format. Displays more host topology information.

***host_name* ...**
Displays information only about the specified host.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Default output

Displays the following fields:

HOSTNAME
Name of the host that runs the RLA.

CPUSET_OS
RLA host operating system.

NCPUS
Total number of CPUs.

FREECPUS
Number of free CPUS.

NNODES
Number of nodes allocated.

NCPU/NODE
Number of CPUs per node.

NSTATIC_CPUSETS
Number of static cpusets allocated.

Long output (-l)

The `-l` option displays a long format listing with the following extra fields:

FREE CPU LIST
List of free CPUs in the Usenet.

```
0-2
```

NFREECPUS ON EACH NODE

Number of free CPUs on each node.

```
2/0,1/1
```

STATIC CPUSETS

List of static cpuset names.

```
NO STATIC CPUSETS
```

CPU_RADIUS

```
2,3,3,3,3,3,3,3
```

- 2 CPUs are available within radius 0
- 3 CPUs are available within radius 1, 2, 3, 4, 5, 6, and 7.

CPUs grouped within a smaller radius can be thought of as being closer together and have better communications performance.

Examples

```
brlainfo hostA hostB hostC
HOSTNAME          CPUSET_OS  NCPUS  NFREECPUS NNODES  NCPU/NODE NSTATIC_CPUSETS
hostA             LINUX_4     2      2         1       2         0
hostB             LINUX_4     4      4         2       2         0
hostC             LINUX_4     4      3         2       2         0

brlainfo -l
HOST: hostC
CPUSET_OS   NCPUS  NFREECPUS NNODES  NCPU/NODE NSTATIC_CPUSETS
LINUX_4       4      3         2       2         0
FREE CPU LIST: 0-2
NFREECPUS ON EACH NODE: 2/0,1/1
STATIC CPUSETS: NO STATIC CPUSETS
CPU_RADIUS: 2,3,3,3,3,3,3,3
```

Chapter 48. brsvadd

Adds an advance reservation.

Synopsis

`brsvadd` [`-o`] [`-f`] [`-d` "*description*"] [`-N` *reservation_name*] [`-nosusp`] [`-q` "*queue_name* ..."] [`-E` *pre_ar_script*] [`-Et` *pre_ar_time*] [`-Ep` *post_ar_script* [`-Ept` *post_ar_time*]]

{`-u` "*user_name* ..." | `-u` "*user_group* ..."}

{[`-unit slot`] `-n` *job_slots* | `-unit host` `-n` *number_hosts*}

{`-m` "*host_name* ... " | "*host_group* ..." [`-R` "*res_req*"] |

[`-m` "*host_name* ... | " *host_group* ..."] `-R` "*res_req*"}

{`-b` *begin_time* `-e` *end_time* | `-t` *time_window*}

`brsvadd` [`-f`] [`-d` "*description*"] [`-N` *reservation_name*] [`-q` "*queue_name* ..."] [`-E` *pre_ar_script*] [`-Et` *pre_ar_time*] [`-Ep` *post_ar_script* [`-Ept` *post_ar_time*]]

`-s` | {`-m` "*host_name* ... | *host_group* ..." [`-R` "*res_req*"] |

[`-m` "*host_name* ..." | `-m` "*host_group* ..."] `-R` "*res_req*"}

{`-b` *begin_time* `-e` *end_time* | `-t` *time_window*}

`brsvadd` [`-o`] [`-f`] `-p` [`-d` "*description*"] [`-N` *reservation_name*]

{`-u` "*user_name* ..." | `-u` "*user_group* ..."}

[`-unit slot` | `-unit host`]

`brsvadd` {`-h` | `-V`}

Description

CAUTION:

By default, this command can be used only by LSF administrators or root.

Reserves job slots or hosts in advance for a specified period for a user or user group, or for system maintenance purposes. Use the `-b` and `-e` options for one-time reservations, and the `-t` option for recurring reservations.

To allow users to create their own advance reservations without administrator intervention, configure advance reservation policies in the `ResourceReservation` section of the `lsb.resources` file.

Only administrators, root, or the users who are listed in the `ResourceReservation` section can add reservations for themselves or any other user or user group.

Advance reservations must be 10 minutes or more in length.

Note:

Advance reservations might be rejected if they overlap other advance reservations that begin or end within a 10-minute time period.

A day is divided into 144 periods. Each period lasts for 10 minutes. For example, `0:0-0:10`, `0:10-0:20`, up to `23:50-24:00`. If the start time or end time of a reservation is in the middle of a time period, LSF reserves the entire period. For example, if one reservation begins at 1:22 and ends at 4:24, a reservation request that starts at 4:25 is rejected because it is within the already reserved 4:20-4:30 time period.

Options

-nosusp
If specified, LSF will not suspend non-advance reservation jobs that are running on the advance reservation hosts when the first advance reservation job starts. Non-advance reservation jobs continue to run, and advance reservation jobs do not start until resources are available. This ensures that resources are not over-committed.

This flag is only valid with user advance reservations.

-o
Creates an open advance reservation. A job with an open advance reservation has the advance reservation property only during the reservation window. After the reservation window closes, the job becomes a normal job, not subject to termination.

An open reservation prevents jobs from being killed if the reservation window is too small. Instead, the job is suspended and normal scheduling policies apply after the reservation window.

-p
Manually creates an advance reservation *placeholder* without a time window or hosts for use by a dynamically scheduled advance reservation. You must use the –u to define a user name or user group that uses the reservation. The **brsvsub** command automatically creates a placeholder and submits a job to the reservation.

-s
Creates a reservation for system use. LSF does not dispatch jobs to the specified hosts while the reservation is active.

When you specify a system reservation with the `-s` option, you do not need to specify the number of job slots to reserve with the `-n` option.

-b *begin_time*
Begin time for a one-time reservation. The begin time has the following form:

```
[[[year:]month:]day:]hour:minute
```

The begin time has the following ranges:

year
Any year after 1900 (*YYYY*).

month
1-12 (*MM*).

day of the month
1-31 (*dd*).

hour
0-23 (*hh*).

minute
0-59 (*mm*).

You must specify at least *hour*:*minute*. Year, month, and day are optional. Three fields are assumed to be *day*:*hour*:*minute*. Four fields are assumed to be *month*:*day*:*hour*:*minute*, and five fields are *year*:*month*:*day*:*hour*:*minute*.

If you do not specify a day, LSF assumes the current day. If you do not specify a month, LSF assumes the current month. If you specify a year, you must specify a month.

The time value for the `-b` option must use the same syntax as the time value for the `-e` option. It must be earlier than the time value for the `-e` option, and it cannot be earlier than the current time.

-d "*description*"
Specifies a description for the reservation to be created. The description must be provided as a double quoted text string. The maximum length is 512 characters.

-E *pre_ar_script*

Specifies the absolute file path to a script that is run to create the advance reservation. If the creator is not root or an LSF administrator, the creator's user group must be an an LSF or queue administrator so that this pre-script can take action on other users' jobs. LSB_START_EBR0KERD=Y must be specified in the `lsf.conf` file for LSF to run the script.

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

The following environment variables are available for use in the script:

AR_NAME
Name of the advance reservation.

AR_QUEUE_LIST
List of queues whose jobs can be run in this advance reservation.

AR_HOST_LIST
List of hosts in this advance reservation. The host is reported even if the advance reservation does not use all slots on the host.

AR_START_TIME
Start time of this advance reservation in epoch seconds.

AR_END_TIME
End time of this advance reservation in epoch seconds.

AR_JOBIDS
The job IDs of jobs that are currently running on this advance reservation's hosts.

AR_CREATOR
Name of the user that created this advance reservation.

AR_OWNERS
Name of the owners of this advance reservation.

The script is run at the start time of the advance reservation unless a pre-time is set with the `-Et` option, then the script is run at the start time minus the specified pre-time. If the script is modified before the script is to be run, the latest version of the script is run at the start time of the script.

The script can use the **bpost** command to notify the job owner that the job was killed by the script. The script can also create its own logs and send notifications to the creator and owner of the advance reservation. LSF does not take any specific action based on the success or failure of the script, and there is no timeout period or action that is associated with this script.

If the conditions of the advance reservation or the job change while the script is running (for example, with the **brsvmod** or **bmod** command), the scripts are not notified and the environment variables do not change. It is the responsibility of the script to handle these changes. In addition, after the script is run, any kill or requeue actions on the jobs cannot be undone if the advance reservation or the job itself is changed with the **brsvmod** or **bmod** command.

-Ep *post_ar_script*

Specifies the absolute file path to a script that is run as the creator of the advance reservation when it expires. If the creator is not root or an LSF administrator, the creator's user group should be an an LSF or queue administrator so that this post-script can take action on other users' jobs. LSB_START_EBR0KERD=Y must be specified in the `lsf.conf` file for LSF to run the script.

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

The following environment variables are available for use in the script:

AR_NAME
Name of the advance reservation.

AR_QUEUE_LIST
List of queues whose jobs can be run in this advance reservation.

AR_HOST_LIST
List of hosts in this advance reservation. The host is reported even if the advance reservation does not use all slots on the host.

AR_START_TIME
Start time of this advance reservation as a UTC time stamp.

AR_END_TIME
End time of this advance reservation as a UTC time stamp.

AR_JOBIDS
The job IDs of jobs that are currently running on this advance reservation's hosts.

AR_CREATOR
Name of the user that created this advance reservation.

AR_OWNERS
Name of the owners of this advance reservation.

The script is run at the expiry time of the advance reservation unless a pre-time is set with the `-Ept` option, then the script is run at the expiry time minus the specified pre-time. If the script is modified before the script is to be run, the latest version of the script is run at the start time of the script.

The script can use the **bpost** command to notify the job owner that the job was killed by the script. The script can also create its own logs and send notifications to the creator and owner of the advance reservation. LSF does not take any specific action based on the success or failure of the script, and there is no timeout period or action that is associated with this script.

If the conditions of the advance reservation or the job change while the script is running (for example, with the **brsvmod** or **bmod** command), the scripts are not notified and the environment variables do not change. It is the responsibility of the script to handle these changes. In addition, after the script is run, any kill or requeue actions on the jobs cannot be undone if the advance reservation or the job itself is changed with the **brsvmod** or **bmod** command.

-Ept ***post_ar_time***
The amount of time, in minutes, before the expiry of the advance reservation for LSF to run the post-script (as specified by the `-Ep` option). This option is ignored if it is specified without the `-Ep` option.

-Et ***pre_ar_time***
The amount of time, in minutes, before the start of the advance reservation for LSF to run the pre-script (as specified by the `-E` option) and to stop dispatching new jobs to the advance reservation hosts.

If this option is specified without the `-E` option, LSF stops dispatching jobs to this advance reservation's hosts at pre-time without running a pre-script.

-e ***end_time***
End time for a one-time reservation. The end time has the following form:

```
[[[year:]month:]day:]hour:minute
```

The end time has the following ranges:

year
Any year after 1900 (*YYYY*).

month
1-12 (*MM*).

day of the month
1-31 (*dd*).

hour
0-23 (*hh*).

minute
0-59 (*mm*).

You must specify at least *hour*:*minute*. Year, month, and day are optional. Three fields are assumed to be *day*:*hour*:*minute*. Four fields are assumed to be *month*:*day*:*hour*:*minute*, and five fields are *year*:*month*:*day*:*hour*:*minute*.

If you do not specify a day, LSF assumes the current day. If you do not specify a month, LSF assumes the current month. If you specify a year, you must specify a month.

The time value for the `-e` option must use the same syntax as the time value for the `-b` option. It must be later than the time value for the `-b` option.

-f

Selects hosts based on the specified resource requirements (`-R`/`-m` option).

Note: If *AR_AVAILABLE_STATUS* in `lsb.params` is defined, then hosts with that status are preferred in the AR creation.

-m "*host_name ... | host_group ...*"

Lists the hosts and groups of hosts that are used for the advance reservation request. At job submission, LSF considers the hosts in the specified order.

If you also specify a resource requirement string with the `-R` option, the `-m` option is not required.

The hosts can be local to the cluster or hosts that are leased from remote clusters.

The number of slots that are specified by the `-n` *<job_slots>* option or hosts that are specified by `-n` *<number_hosts>* option must be less than or equal to the actual number of hosts that are specified by the `-m` option.

Note: When you use the `-m` option to specify multiple hosts for advance reservation, some hosts might not be selected for advance reservation (for example, because the hosts are exclusive and in `closed` status). If at least one host in the list was successfully selected for advance reservation, the **brsrvadd** command indicates that the advance reservation was successfully created.

-N *reservation_name*

Specifies a user-defined advance reservation name unique in an LSF cluster. The name is a string of letters, numeric characters, underscores, and dashes. The name must begin with a letter. The maximum length of the name is 40 characters.

If no user-defined advance reservation name is specified, LSF creates the reservation with a system assigned name in the following form:

```
user_name#sequence
```

In the following example, the **brsvadd** command has no `-N` option, so the reservation is created with the system assigned name `Reservation user2#0`:

```
brsvadd -n 3 -m "hostA hostB" -u user2 -b 16:0 -e 17:0 -d "Production AR test"
Reservation user2#0 (Production AR test) is created
```

In the following example, the **brsvadd** command specifies the name `Production_AR` on the `-N` option, so the reservation is created with the specified name:

```
brsvadd -n 2 -N Production_AR -m hostA -u user2 -b 16:0 -e 17:0 -d "Production AR test"
Reservation Production_AR (Production AR test) is created
```

If a job already references a reservation with the specified name, an error message is returned: The `specified reservation name is referenced by a job`.

-n *job_slots* or *number_hosts*

The number of either job slots or hosts (specified by the `-unit` option) to reserve. For a slot-based advance reservation (`brsvadd -unit slot`), the `-n` option specifies the total number of job slots to reserve. For host-based advance reservation (`brsvadd -unit host`), the `-n` option specifies the total number of hosts to reserve.

The *job_slots* or *number_hosts* value must be less than or equal to the actual number of slots or hosts that are selected by the `-m` or `-R` option.

If you also specify the reservation for system use with the `-s` option, the `-n` is not required.

-q "*queue_name ...*"
Specifies the queues whose jobs are allowed to run on the advance reservation hosts even if the jobs' run limits are greater than the amount of time until the advance reservation starts.

-R "*res_req*"
Selects hosts for the reservation according to the specified resource requirements. Only hosts that satisfy the resource requirement expression are reserved. The `-R` option accepts any valid resource requirement string, but only the `select` and `same` strings take effect.

If you also specify a host list with the `-m` option, the `-R` is not required.

For more information about specifying resource requirement strings, see *Administering IBM Spectrum LSF*.

-t *time_window*
Time window for a recurring reservation.

To specify a time window, specify two time values that are separated by a hyphen (`-`), with no space in between:

```
time_window = begin_time-end_time
```

Times are specified in the following format:

```
[day:]hour[:minute]
```

All fields are numbers with the following ranges:

day of the week
0-6 (0 is Sunday).

hour
0-23

minute
0-59

Specify a time window one of the following ways:

- *hour-hour*
- *hour*:*minute*-*hour*:*minute*
- *day*:*hour*:*minute*-*day*:*hour*:*minute*

The default value for minute is 0 (on the hour). The default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute, LSF assumes the first minute of the hour (`:00`). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

To prevent running jobs from being killed when the reservation expires, LSF administrators can use the **bmod -t** option to change the termination time of the job before the reservation window closes.

When the job starts running, the run limit of the reservation is set to the minimum of the job run limit (if specified), the queue run limit (if specified), or the duration of the time window.

-u "*user_name ...*" | *"user_group ...*"
A list of users and user groups that have permission to use advance reservation.

The `-u "user_name ... | user_group ..."` option does not support the `@cluster` notation for advance reservations on remote clusters.

-unit [slot | host]

Specifies whether an advance reservation is for a number of slots or hosts. If the `-unit` option is not specified, the advance reservation request uses the slot unit by default.

The following options are required when used with the **brsvadd** command, regardless of whether you use the slot or host unit:

- The number of slots or hosts to reserve, with the `-n` option.
- The list of candidate hosts, with the `-m` option, the `-R` option, or both.
- Users or user groups that have permission to use the advance reservation, with the `-u` option.
- A time period for the reservation, with either the `-t` or the `-b` option and the `-e` option together.

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

Examples

The following command creates a one-time advance reservation for 14 job slots on hosts `hostA` and `hostB` for `user1` and `group1` between 6:00 AM and 8:00 AM today:

```
brsvadd -unit slot -n 14 -m "hostA hostB" -u "user1 group1" -b 6:0 -e 8:0
Reservation "user1#0" is created
```

The following command creates an advance reservation for four hosts and the reserved hosts have at least 16 slots:

```
brsvadd -unit host -n 4 -R "maxslots>=16"" -u "groupA groupB groupC" -b 3:0 -e 4:0
```

```
Reservation "groupA#0" is created
```

The following command creates an open advance reservation for 1024 job slots on host `hostA` for user `user1` between 6:00 AM and 8:00 AM today.

```
brsvadd -o -n 1024 -m hostA -u user1 -b 6:0 -e 8:0
Reservation "user1#0" is created
```

See also

brsvdel, **brsvmod**, **brsvs**, `lsb.resources`

Chapter 49. brsvdel

Deletes an advance reservation.

Synopsis

`brsvdel` *reservation_ID* ...

`brsvdel {-h | -V | -f}`

Description

CAUTION:

By default, this command can be used only by LSF administrators or root.

Deletes advance reservations for the specified reservation IDs.

For example, the following command was used to create the reservation `user1#0`,

```
brsvadd -n 1024 -m hostA -u user1 -b 13:0 -e 18:0
Reservation "user1#0" is created
```

The following command deletes the reservation:

```
brsvdel user1#0
Reservation user1#0 is being deleted
```

You can delete multiple reservations at a time.

To allow users to delete their own advance reservations without administrator intervention, configure advance reservation policies in the `ResourceReservation` section of the `lsb.resources` file.

Administrators and root can delete any reservations. Users who are listed in the `ResourceReservation` section can delete only reservations that they created themselves.

Options

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

-f

Forces deletion of an AR. The running jobs will be detached and treated as regular jobs.

If **LSB_DISABLE_SUSPEND_AR_JOBS** under `lsf.conf` is set to Y, then these jobs will continue running and other regular suspending jobs will continue to be suspended. If it is set to N, these jobs will be suspended and will compete with other regular jobs for slots.

See also

brsvadd, **brsvmod**, **brsvs**, `lsb.resources`

Chapter 50. brsvjob

Shows information about jobs submitted with the **brsvsub** command to a specific advance reservation.

Synopsis

brsvjob *reservation_name*

brsvjob [-h]

Example

```
brsvjob user1#0

Job <1>, User <user1>, Project <default>, Status <DONE>, Queue <normal>, Comm
                     and <lsfrsv -N user1#0 -D 10 -n 2>, Share group charged
                     </user1>, Job Description <user1#0>
Tue Jun  6 21:47:58: Submitted from host <hostA>, CWD
                      </scratch/dev/user1/lsf>, 2 Task(s);

RUNLIMIT
11.0 min of hostA
Tue Jun  6 21:47:58: Started 2 Task(s) on Host(s) <hostA>
                     <hostA>, Allocated 2 Slot(s) on Host(s)
                     <hostA> <hostA>, Execution Home </home/user1>, Ex
                     ecution CWD </scratch/dev/user1/lsf>;
Tue Jun  6 21:47:58: Done successfully. The CPU time used is 0.1 seconds.

SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
loadSched   -     -     -     -       -     -    -     -     -      -      -
loadStop    -     -     -     -       -     -    -     -     -      -      -

RESOURCE REQUIREMENT DETAILS:
Combined: select[type == local] order[r15s:pg]
Effective: select[type == local] order[r15s:pg]
```

Chapter 51. brsvmod

Modifies an advance reservation.

Synopsis

`brsvmod` [`-o` | `-on`] [`-d` "*description*"] [`-u` *"user_name ..."* | *"user_group ..."*] [`-nosusp` | `-nonsuspn`] [`-q` "*queue_name ...*" | `-qn`] [`-E` *pre_ar_script* | `-En`] [`-Et` *pre_ar_time* | `-Etn`] [`-Ep` *post_ar_script* | `-Epn`] [`-Ept` *post_ar_time* | `-Eptn`] [[`-b` *begin_time* | [+|-]*minutes*] [`-e` *end_time* | [+|-]*minutes*]] | [`-t` *time_window*] *reservation_ID*

`brsvmod addhost` {`-n` *number_unit* `-R` "*res_req*" [`-m` "*host_name ...* | *host_group ...*"]} | {[`-n` *number_unit*] `-m` "*host_name ...* | *host_group ...*"} *reservation_ID*

`brsvmod adduser -u` *"user_name ..."* | "*user_group ...*" *reservation_ID*

`brsvmod disable` {`-td` "*begin_date-end_date*" | `-tn`} [`-f`] *reservation_ID*

`brsvmod rmhost` {`-n` *number_unit* [`-m` "*host_name ...* | *host_group ...*"]} | {[`-n` *number_unit*] `-m` "*host_name ...* | *host_group ...*"} *reservation_ID*

`brsvmod rmuser -u` *"user_name ..."* | "*user_group ...*"] *reservation_ID*

`brsvmod` {`-h` | `-V`}

Description

Important:

By default, this command can be used only by LSF administrators or root.

Replaces advance reservation option values previously created, extends or reduces the reservation time window, or adds or removes reserved hosts of the advance reservation that is specified by *reservation_ID*. For a recurring reservation, can disable specified occurrences of the reservation.

Administrators and root can modify any reservations. Users who are listed in the `ResourceReservation` section of the `lsb.resources` file can modify only reservations that they created themselves.

The original value for user, user group, or time window, can be overridden with a new value by specifying the option as in **brsvadd**. Change a reservation from closed (the default) to open with the `-o` option, or from open to `closed` with the `-on` option. You can also use the subcommands **adduser** and **rmuser** to add or remove users and user groups that are assigned to the advance reservation.

Options `-n`, `-m`, and `-R` must be used with the subcommands **addhost** or **rmhost**. These options allow adding or removing from the original values.

The `-td` and `-tn` options are only allowed in the **disable** subcommand.

All subcommands are mutually exclusive. The time window options `-b`, `-e`, and `-t` are not valid in any of the subcommands.

You cannot modify the start time of an active reservation.

The **brsvmod** command does not support the *reservation_ID*`@`*cluster_name* notation for advance reservations on remote clusters, or the *user_name*`@`*cluster_name* notation for reservations with remote users.

The *number_unit* requirement of the `-n` option must be satisfied. The `-m` or `-R` option provides a candidate list for processing, and triggers an error only if no valid hosts are in the list. For instance, the following option requires three slots:

```
-n 3 -m "host1 host2"
```

LSF tries to find as many slots as possible from host1. If three slots are not available on host1, LSF tries to find the rest from host2. Hosts with no slots available are removed from the list when the request is handled.

If you do not use the **brsvsub** command to create a dynamically scheduled reservation, you can manually add a time window to a reservation placeholder that was created with the **brsvadd -p** option. Use the brsvmod -b *begin_time* -e *end_time reservation_ID* command.

To add resources to a placeholder, use the **brsvmod addhost** command.

- By default, a placeholder reservation is a one-time reservation. You can't change a placeholder to a recurring reservation.
- A placeholder reservation with a time window is cleaned when the reservation expires.

Subcommands

addhost {-n *number_unit* -R "*res_req*" [-m "*host_name ... | host_group ...*"]} | {[-n *number_unit*] -m "*host_name ... | host_group ...*"} *reservation_ID*

Adds hosts and slots on hosts into the original reservation allocation. The hosts can be local to the cluster or hosts that are leased from remote clusters.

Adding a host without the -n option reserves all available hosts or slots on the host that are not already reserved by other reservations. The -n cannot be used alone. You can specify the number of slots to be added from the host list that is specified with the -n option. The -m option can be used alone if no host group is specified in the list. If you specify the -R option, you must also specify the -n option.

The specified number of units (slots or hosts) must be less than or equal to the available number of slots for the hosts or hosts themselves.

Restriction: Only hosts can be added (with the -m option) to a system reservation. Slots cannot be added (with the -n option) to a system reservation.

adduser -u "*user_name ... | user_group ...*" *reservation_ID*

Adds users and user groups to an advance reservation.

disable {-td "*begin_date-end_date*" | -tn} [-f] *reservation_ID*

Disables specified periods, or instances, of a recurring advance reservation. The *start_date* and *end_date* represent the start and end date of a period in which the reservation is disabled. These periods must take one of the following forms:

- yyyy:mm:dd-yyyy:mm:dd
- mm:dd-mm:dd - the current year is assumed
- dd-dd - the current month and year are assumed

The start date must be the same as or earlier than the end date.

If a reservation is disabled for a specified day, then it does not become active on that day, and remains inactive during the reservation time window. Non-recurring reservations are able to use slots of the recurring reservation during the time window. The -tn option is a shortcut that disables a reservation on the starting day of the next instance of the reservation time window; that is, the instance that starts nearest in the future. If the reservation is disabled for this day, the modification request is rejected.

For example, for a weekly reservation with time window from Wednesday 9 AM to Friday 10 PM, if the current day is Monday, then running the command with the -tn option disables the reservation from Wednesday to Friday of the current week. However, if the current day is Thursday, then the reservation is disabled from Wednesday to Friday of the following week. If it is Wednesday, then whether to disable in the current week or following week depends on whether the start time of the instance is passed. If not, then the reservation is disabled in the current week, otherwise the following week's reservation is disabled.

Running the disable command with the `-t`n option twice on Monday tries to disable twice in the current week. The second run has no effect, but is rejected because the specified reservation instance is already disabled.

After a reservation is disabled for a period, it cannot be enabled; that is, the disabled periods remain fixed. Before a reservation is disabled, you are prompted to confirm whether to continue disabling the reservation. Use the `-f` option to silently force the command to run without prompting for confirmation; for example, to allow for automating disabling reservations from a script.

rmhost {-n *number_unit* [-m "*host_name* ... | *host_group* ..."]} | {[-n *number_unit*]-m "*host_name* ... | *host_group* ..."} *reservation_ID*

Removes hosts or slots on hosts from the original reservation allocation. You must specify either the `-n` or `-m` option. Use the `-n` option to specify the number of hosts to be released or slots to be released from reserved hosts. Removing a host without the `-n` option releases all hosts or reserved free slots on the host. The specified number of units (slots or hosts) must be less than or equal to the available number of hosts or slots for the hosts.

You can remove only a whole host from a system reservation.

How many slots or hosts can be removed depends on the number of slots that are free while the reservation is active. The **rmhost** subcommand cannot remove more slots than are free on the host on both one-time and recurring reservations that are active. If you want to remove more slots from the reservation, you must wait until running jobs finish or the reservation is inactive.

rmuser -u "*user_name* ... | *user_group* ..." *reservation_ID*

Removes users and user groups from an advance reservation.

Options

-nosusp | -nosuspn

If specified, LSF no longer suspends non-advance reservation jobs that are running on the advance reservation hosts when the first advance reservation job starts. Non-advance reservation jobs continue to run, and advance reservation jobs do not start until resource are available. This ensures that resources are not over-committed. If the `-nosuspn` option is specified, LSF suspends non-advance reservation jobs that are running on the advance reservation hosts when the first advance reservation job starts.

This flag is only valid with user advance reservations if the advance reservation is inactive and not within the pre-time period.

-o

Changes a closed advance reservation to open, or cancels an open reservation.

If the reservation is open, all jobs in the reservation become normal jobs, not subject to termination when the reservation window closes. The `-on` option closes the reservation when it expires. The running jobs of an open reservation are terminated when the reservation is changed into closed. The termination times of running jobs of a closed reservation are removed if the reservation is changed to open. The termination time of running jobs is set by the **mbatchd** daemon but checked by the **sbatchd** daemon. Termination time is an absolute time based on master host, so all hosts in the cluster must be synchronized with the local time on the master host. If the **sbatchd** daemon and the **mbatchd** daemon are not synchronized, termination might not occur at the correct time.

-b *begin_time* | [+ | -]*minutes*

Replaces the begin time for a one-time reservation, or gives an offset in minutes to the current begin time.

Restriction: You cannot modify the begin time of an active reservation.

The begin time is in the following form:

```
[[[year:]month:]day:]hour:minute
```

The begin time has the following ranges:

year
Any year after 1900 (YYYY).

month
1-12 (MM).

day of the month
1-31 (dd).

hour
0-23 (hh).

minute
0-59 (mm).

Year, month, and day are optional. You must specify at least *hour*:*minute*:

- Three fields are assumed to be *day*:*hour*:*minute*
- Four fields are assumed to be *month*:*day*:*hour*:*minute*
- Five fields are *year*:*month*:*day*:*hour*:*minute*

If you do not specify a day, LSF assumes the current day. If you do not specify a month, LSF assumes the current month. If you specify a year, you must specify a month.

The offset is in minutes, an integer with a prefix + or -. For example, **-b+5** moves the begin time 5 minutes later, and **-b-5** moves the begin time 5 minutes earlier.

The modified time value for the begin time (-b option) must use the same syntax as the time value for the end time (-e option). The begin time must be earlier than the time value for the end time. The begin time cannot be earlier than the current time.

-d "*description*"
Replaces or sets a description for the reservation. The description must be provided as a double quoted text string. The maximum length is 512 characters.

-E *pre_ar_script* | -En
Replaces the absolute file path to the script that is run to create the advance reservation. The -En option removes the script so that no scripts are run. If the creator is not root or an LSF administrator, the creator's user group should be an an LSF or queue administrator so that this pre-script can take action on other users' jobs. LSB_START_EBROKERD=Y must be specified in the lsf.conf file for LSF to run the script.

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

The following environment variables are available for use in the script:

AR_NAME
Name of the advance reservation.

AR_QUEUE_LIST
List of queues whose jobs can be run in this advance reservation.

AR_HOST_LIST
List of hosts in this advance reservation. The host is reported even if the advance reservation does not use all slots on the host.

AR_START_TIME
Start time of this advance reservation as a UTC time stamp.

AR_END_TIME
End time of this advance reservation as a UTC time stamp.

AR_JOBIDS
The job IDs of jobs that are currently running on this advance reservation's hosts.

AR_CREATOR
Name of the user that created this advance reservation.

AR_OWNERS
Name of the owners of this advance reservation.

The script is run at the start time of the advance reservation unless a pre-time is set with the `-Et` option, then the script is run at the start time minus the specified pre-time. If the script is modified before the script is to be run, the latest version of the script is run at the start time of the script.

When the AR_END_TIME is changed, the pending jobs should not be dispatched if the specified runlimit exceeds the end time of the AR.

The script can use the **bpost** command to notify the job owner that the job was killed by the script. The script can also create its own logs and send notifications to the creator and owner of the advance reservation. LSF does not take any specific action based on the success or failure of the script, and there is no timeout period or action that is associated with this script.

If the conditions of the advance reservation or the job change while the script is running (for example, with the **brsvmod** or **bmod** command), the scripts are not notified and the environment variables do not change. It is the responsibility of the script to handle these changes. In addition, after the script is run, any kill or requeue actions on the jobs cannot be undone if the advance reservation or the job itself is changed with the **brsvmod** or **bmod** command.

-Ep *post_ar_script* **| -Epn**
Replaces the absolute file path to the script that is run as the creator of the advance reservation when it expires. The `-En` option removes the script so that no scripts are run. If the creator is not root or an LSF administrator, the creator's user group should be an an LSF or queue administrator so that this post-script can take action on other users' jobs. LSB_START_EBROKERD=Y must be specified in the `lsf.conf` file for LSF to run the script.

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

The following environment variables are available for use in the script:

AR_NAME
Name of the advance reservation.

AR_QUEUE_LIST
List of queues whose jobs can be run in this advance reservation.

AR_HOST_LIST
List of hosts in this advance reservation. The host is reported even if the advance reservation does not use all slots on the host.

AR_START_TIME
Start time of this advance reservation as a UTC time stamp.

AR_END_TIME
End time of this advance reservation as a UTC time stamp.

AR_JOBIDS
The job IDs of jobs that are currently running on this advance reservation's hosts.

AR_CREATOR
Name of the user that created this advance reservation.

AR_OWNERS
Name of the owners of this advance reservation.

The script is run at the expiry time of the advance reservation unless a pre-time is set with the `-Ept` option, then the script is run at the expiry time minus the specified pre-time. If the script is modified before the script is to be run, the latest version of the script is run at the start time of the script.

When the AR_END_TIME is changed, the pending jobs should not be dispatched if the specified runlimit exceeds the end time of the AR.

The script can use the **bpost** command to notify the job owner that the job was killed by the script. The script can also create its own logs and send notifications to the creator and owner of the advance

reservation. LSF does not take any specific action based on the success or failure of the script, and there is no timeout period or action that is associated with this script.

If the conditions of the advance reservation or the job change while the script is running (for example, with the **brsvmod** or **bmod** command), the scripts are not notified and the environment variables do not change. It is the responsibility of the script to handle these changes. In addition, after the script is run, any kill or requeue actions on the jobs cannot be undone if the advance reservation or the job itself is changed with the **brsvmod** or **bmod** command.

-Ept *post_ar_time* | -Eptn
Changes the amount of time, in minutes, before the expiry of the advance reservation for LSF to run the post-script (as specified by the -Ep option). The -Ept option is ignored if the -Ep option is not enabled.

If you specify the -Eptn option, the post-script is run at the expiry time of the advance reservation

-Et *pre_ar_script* | -Etn
Changes the amount of time, in minutes, before the start of the advance reservation for LSF to run the pre-script (as specified by the -E option) and to stop dispatching new jobs to the advance reservation hosts.

If the -E option is not enabled, specifying the -Et option means that LSF stops dispatching jobs to this advance reservation's hosts at pre-time without running a pre-script.

If you specify the -Etn option, the pre-script is run at the start time of the advance reservation

-e *end_time* | [+ | -]*minutes*
Replaces the end time for a one-time reservation, or gives an offset in minutes to the current end time.

By giving a positive offset to the end time, you extend the duration of a reservation so that the jobs in the reservation can run longer. Shrinking the reservation with a negative value terminates running jobs earlier.

The end time is in the following form:

```
[[[year:]month:]day:]hour:minute
```

The end time has the following ranges:

year
Any year after 1900 (YYYY).

month
1-12 (MM).

day of the month
1-31 (dd).

hour
0-23 (hh).

minute
0-59 (mm).

Year, month, and day are optional. You must specify at least *hour*:*minute*:

- Three fields are assumed to be *day*:*hour*:*minute*
- Four fields are assumed to be *month*:*day*:*hour*:*minute*
- Five fields are *year*:*month*:*day*:*hour*:*minute*

If you do not specify a day, LSF assumes the current day. If you do not specify a month, LSF assumes the current month. If you specify a year, you must specify a month.

The time value for the end time (-e option) must use the same syntax as the time value for the (-b option). The end time must be later than the time value for the begin time.

-m "*host_name*... | *host_group*..."
Changes the list of hosts for which job slots or number of hosts that are specified with the `-n` option are reserved. At job submission, LSF uses the hosts in the specified order.

If you also specify a resource requirement string with the `-R` option, the `-m` option is not required.

The hosts can be local to the cluster or hosts that are leased from remote clusters.

-n *number_unit*
Changes the number of either job slots or hosts to reserve (based on the unit that is specified by the `brsvadd -unit slot | host` command. The *number_unit* variable must be less than or equal to the actual number of slots for the hosts that are selected by the `-m` or `-R` option for the reservation.

If you also specify the reservation for system, use the `-n` option with the `-s` option. The `-n` option is not required.

-q "*queue_name* ..." | -qn
Changes the queues whose jobs are allowed to run on the advance reservation hosts even if the jobs' run limits are greater than the amount of time until the advance reservation starts. The `-qn` option removes the list of allowed queues.

-R "*res_req*"
Changes the host selection for the reservation according to the specified resource requirements. Only hosts that satisfy the resource requirement expression are reserved. The `-R` option accepts any valid resource requirement string, but only the `select` and `same` strings take effect.

If you also specify a host list with the `-m` option, the `-R` option is not required.

For more information about specifying resource requirement strings, see *Administering IBM Spectrum LSF*.

-t *time_window*
Replaces the time window with a new one to shift a recurring reservation. You cannot modify the start time of a recurring reservation that has current active instance.

To specify a time window, specify two time values. Separate the time values by a hyphen (`-`) with no space in between:

```
time_window = begin_time-end_time
```

Times are specified in the following format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

day of the week
0-6 (0 is Sunday).

hour
0-23

minute
0-59

Specify a time window:

- *hour-hour*
- *hour*:*minute*-*hour*:*minute*
- *day*:*hour*:*minute*-*day*:*hour*:*minute*

The default value for minute is 0 (on the hour). The default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute, LSF assumes the first minute of the hour (`:00`). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

LSF administrators can prevent running jobs from being killed when the reservation expires by changing the termination time of the job that uses the reservation with the **bmod -t** command before the reservation window closes.

When the job starts running, the run limit of the reservation is set to the minimum of the job run limit (if specified), the queue run limit (if specified), or the duration of the time window.

-u "*user_name*... | *user_group* ..."
Replaces the list of users or groups who are able to submit jobs to a reservation. Replacing the list of users or groups does not affect the currently running jobs.

Jobs that are submitted by the original users or groups to the reservation still belong to the reservation and scheduled as advance reservation jobs, but newly submitted jobs from the users or groups that were removed from the reservation cannot use the reservation any longer.

The `-u "user_name ... | user_group ..."` option does not support the `@cluster` notation for advance reservations on remote clusters.

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

Examples

The following command adds a host to an existing reservation.

```
brsvmod addhost -m hostB user1#0
Reservation user1#0 is modified
```

The following example disables the advance reservation between January 1 and January 6, 2008, inclusive.

```
brsvmod disable {-td "2008:01:01-2008:01:06"}
```

See also

brsvadd, **brsvdel**, **brsvs**, `lsb.resources`

Chapter 52. brsvs

Displays advance reservations.

Synopsis

`brsvs [-l | -w] [-p all | -p "host_name ..."]`

`brsvs [-l | -w] [-z all | -z "host_name ..."]`

`brsvs [-l | -w] reservation_name`

`brsvs [-c all | -c "policy_name"]`

`brsvs [-h | -V]`

Description

By default, displays the current advance reservations for all hosts, users, and groups.

You can also see advance reservations across clusters:

- The default `all` option includes both local and remote reservations.
- *host_name* does not take the *host_name*`@`*cluster_name* argument.

By default, the **brsvs** command truncates the reservation ID (RSVID) at 11 characters. Use the `-w` option to see the full reservation ID.

Options

-l
Displays advance reservations in a long multiline format. In addition to the standard output, the `-l` option displays queues that can use the advance reservation hosts before the advance reservation starts, the pre-script, pre-script time, post-script, post-script time, the reservation type (open or `closed`), whether non-advance reservation jobs are allowed to continue running after the first advance reservation starts, and the job IDs of any jobs that are associated with the specified advance reservation. Reservations are sorted by status.

The NCPUS field displays real-time advance reservation usage, in the format: used slots/total slots.

-w
Wide format. Displays reservation information without truncating fields.

-c all | -c "*policy_name* ..."
Shows advance reservation policies that are defined in the `lsb.resources` file. By default, displays all policy names.

The `all` keyword shows detailed information for all policies.

-p all | -p "*host_name* ..."
Shows a weekly planner for specified hosts that use advance reservations.

The `all` keyword shows a weekly planner for all hosts with reservations.

-z all | -z "*host_name*"
Shows a planner with only the weekly items that have reservation configurations. Empty lines are omitted.

The `all` keyword shows a weekly planner for all hosts with reservations.

-h
Prints command usage and exits.

-V
Prints LSF release version and exits.

Output

RSVID
The advance reservation ID.

TIME_WINDOW
Time window for a recurring reservation.

Values are separated by a hyphen (-), with no space in between:

```
time_window = begin_time-end_time
```

Times are specified in the following format:

```
[day:]hour[:minute]
```

All fields are numbers with the following ranges:

day of the week
0-6 (0 is Sunday).

hour
0-23.

minute
0-59.

A time window can be specified in any of the following ways:

- *hour-hour*
- *hour*:*minute-hour*:*minute*
- *day*:*hour*:*minute-day*:*hour*:*minute*

The default value for minute is 0 (on the hour). The default value for day is every day of the week.

USER
The list of users and user groups that are specified for the advance reservation.

Examples

The following command shows a reservation with the name `reservation1`

```
brsvs -c reservation1
Policy Name: reservation1
Users: ugroup1 ~user1
Hosts: hostA hostB
Time Window: 8:00-13:00
```

The following example shows a reservation placeholder for a dynamically scheduled reservation. The resources and time window are blank until a job actually uses the reservation:

```
brsvs user2#20
RSVID        TYPE      USER       NCPUS          RSV_HOSTS     TIME_WINDOW
user2#20     user     user2       0/0             none:0/0      -
```

See also

brsvadd, **brsvmod**, **brsvdel**, `lsb.resources`

Chapter 53. brsvsub

Creates a dynamically scheduled reservation and submits a job to fill the advance reservation when the resources required by the job are available.

Synopsis

`brsvsub` [`-I`] [`-D` *duration*] [`-o`] [`-q` *queue_name*] [`-unit slot`] `-n` *job_slots* | `-unit host -n` *number_hosts*

`-u` "*user_name* ..." | `-u` "*user_group* ..."}

[`-m` "*host_name* ... " | " *host_group* ..."] [`-R` "*res_req*"] |

[`-N` *reservation_name*]

`brsvsub -h`

Description

Specify the requirements of the reservation: user name, reservation duration, type of reservation, queue, and hosts required for the reservation.

Options

-D *duration*
Sets a duration for the reservation in minutes.

-I
Specifies an interactive job.

-o
Creates an open advance reservation. A job with an open advance reservation has the advance reservation property only during the reservation window. After the reservation window closes, the job becomes a normal job, not subject to termination.

An open reservation prevents jobs from being killed if the reservation window is too small. Instead, the job is suspended and normal scheduling policies apply after the reservation window.

-m "*host_name ... | host_group ...*"
Lists the hosts and groups of hosts that are used for the reservation request. At job submission, LSF considers the hosts in the specified order.

The hosts can be either local to the cluster or leased from remote clusters.

The number of slots that are specified by the `-n` *job_slots* option or the number of hosts that are specified by the `-n` *number_hosts* option must be less than or equal to the actual number of hosts that are specified by the `-m` option.

Note: When you use the `-m` option to specify multiple hosts for the reservation, some hosts might not be selected (for example, because the hosts are exclusive and in `closed` status).

-N *reservation_name*
Specifies a user-defined reservation name unique in an LSF cluster. The name is a string of letters, numeric characters, underscores, and dashes. The name must begin with a letter. The maximum length of the name is 40 characters.

If no user-defined reservation name is specified, LSF creates the reservation with a system assigned name in the following form:

```
user_name#sequence
```

If a job already references a reservation with the specified name, an error message is returned: The `specified reservation name is referenced by a job`.

-n *job_slots* or *number_hosts*
The number of either job slots or hosts (specified by the `-unit` option) to reserve. For a slot-based reservation (`-unit slot`), the `-n` option specifies the total number of job slots to reserve. For host-based reservation (`-unit host`), the `-n` option specifies the total number of hosts to reserve.

The *job_slots* or *number_hosts* value must be less than or equal to the actual number of slots or hosts that are selected by the `-m` or `-R` option.

-q *queue_name*
Specifies the queue for the submitted job that uses the dynamically scheduled reservation.

-R "*res_req*"
Selects hosts for the reservation according to the specified resource requirements. Only hosts that satisfy the resource requirement expression are reserved. The `-R` option accepts any valid resource requirement string, but only the `select` string takes effect.

If you also specify a host list with the `-m` option, the `-R` is not required.

For more information about specifying resource requirement strings, see *Administering IBM Spectrum LSF*.

-u "*user_name* ..." | *"user_group* ..."
A list of users and user groups that have permission to use the dynamically scheduled reservation.

The `-u "user_name ... | user_group ..."` option does not support the `@cluster` notation for advance reservations on remote clusters.

-unit [slot|host]
Specifies whether the reservation is for a number of slots or hosts. By default, if the `-unit` option is not specified, the reservation request uses the slot unit by default.

Regardless of whether you use the slot or host unit, the following options are required when used with the **brsvsub** command:

- The number of slots or hosts to reserve, with the `-n` option.
- Users or user groups that have permission to use the advance reservation, with the `-u` option.

-h
Prints command usage and exits.

Example

Submit a dynamically scheduled advance reservation:

```
brsvsub -D 10 -n 2 -unit host -u user1
Placeholder advance reservation user1#19 is being scheduled by job <28> in the default queue
<normal>.
```

Use the **brsvs** command to query the scheduled advance reservation:

```
brsvs -l user1#19
RSVID        TYPE      USER       NCPUS          RSV_HOSTS     TIME_WINDOW
user1#19     user     user1       0/16           hostA:0/8    4/20/19/3-4/20/19/13
                                                 hostB:0/8
Reservation Status: Active
Description: job <28>
Creator: user1
Reservation Type: CLOSED
Resource Unit: Host
```

Use the **brsvjob** command to see information about jobs submitted with the **brsvsub** command.

```
brsvjob user1#19

Job <28>, User <user1>, Project <default>, Status <DONE>, Queue <normal>, Comm
                     and <lsfrsv -N user1#19 -D 10 -n 2>, Share group charged
                     </user1>, Job Description <user1#19>
Tue Jun  6 21:47:58: Submitted from host <hostA>, CWD
                      </scratch/dev/user1/lsf>, 2 Task(s);
```

```
RUNLIMIT
11.0 min of hostA
Tue Jun  6 21:47:58: Started 2 Task(s) on Host(s) <hostA>
                     <hostB>, Allocated 2 Slot(s) on Host(s)
                     <hostA> <hostB>, Execution Home </home/user1>, Ex
                     ecution CWD </scratch/dev/user1/lsf>;
Tue Jun  6 21:47:58: Done successfully. The CPU time used is 0.1 seconds.

SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
loadSched   -     -     -     -       -     -    -     -     -      -      -
loadStop    -     -     -     -       -     -    -     -     -      -      -

RESOURCE REQUIREMENT DETAILS:
Combined: select[type == local] order[r15s:pg]
Effective: select[type == local] order[r15s:pg]
```

See also

brsvadd, **brsvs**, **brsvmod**, **brsvdel**, lsb.resources

Chapter 54. brun

Forces a job to run immediately.

Synopsis

`brun [-b] [-c] [-f] -m "host_name[#num_cpus] ... " | "cluster_name" job_ID`

`brun [-b] [-c] [-f] -m "host_name[#num_cpus] ... " | "cluster_name" "job_ID[index_list]"`

`brun [-h | -V]`

Description

Important:

Only administrators can use the **brun** command. In LSF multicluster capability job forwarding model, you can only run the **brun** command from the submission cluster.

Forces a pending or finished job to run immediately on specified hosts.

In the LSF multicluster capability job forwarding model, the **brun -m** command forces a pending or finished job to run on local hosts. See *Using IBM Spectrum LSF multicluster capability* for details.

In IBM Spectrum LSF Advanced Edition, the **brun -m** command forces a pending or finished job to be forwarded to a remote execution cluster. See *Using IBM Spectrum LSF Advanced Edition* for details.

A job that has been forced to run is counted as a running job. Forcing the job to run might violate the user, queue, or host job limits, and fairshare priorities. The forced job can run on hosts with an exclusive resource definition.

A job that has been forced to run cannot be preempted by other jobs even if it is submitted to a preemptable queue and other jobs are submitted to a preemptive queue.

By default, after the job is started, it is still subject to run windows and suspending conditions.

LSF administrators can use the **brun** command to force jobs with an advance reservation to run before the reservation is active, but the job must finish running before the time window of the reservation expires.

For example, if the administrator forces a job with a reservation to run one hour before the reservation is active, and the reservation period is 3 hours, a 4 hour run limit takes effect.

Options

-b

Causes a checkpointable job to start over from the beginning, as if it had never been checkpointed.

-c

Distribute job slots for a multihost parallel job according to free CPUs.

By default, if a parallel job spans for more than one host, LSF distributes the slots based on the static CPU counts of each host listed in the `-m` option. Use the `-c` option to distribute the slots based on the free CPUs of each host instead of the static CPUs.

The `-c` option can be only applied to hosts whose total slot counts equal to their total CPU counts. The **MXJ** parameter in the `lsb.hosts` file must be less than or equal to the number of CPUs and the `PJOB_LIMIT=1` parameter must be specified in the queue (the `lsb.queues` file).

For example, a 6-CPU job is submitted to `hostA` and `hostB` with 4 CPUs each. Without the `-c` option, LSF lets the job take 4 slots from `hostA` first and then take 2 slots from `hostB` regardless to the status or the slots usage on `hostA` and `hostB`. If any slots on `hostA` are used, the job remains pending. With the `-c` opton, LSF takes into consideration that `hostA` has 2 slots in use and `hostB` is

completely free, so LSF is able to dispatch the job using the 2 free slots on hostA and all 4 slots on hostB.

-f
Forces the job to run without being suspended due to run windows or suspending conditions.

-m "*host_name*[*#num_cpus*] ... " | *"cluster_name*"
Required. Specify one or more hosts on which to run the job.

You can optionally specify the number of CPUs required per host for multihost parallel jobs. The *#num_cpus* option distributes job slots according the number of CPUs on the host. If the *#num_cpus* option is not defined, or if the *#num_cpus* option is greater than the number of static CPUs on the host (or the number of free CPUs if -c is specified), LSF distributes job slots according to the number of static CPUs on the host. If the *#num_cpus* option is specified, LSF distributes job slots according to the number of free CPUs on the host. The number sign (*#*) is required as a prefix to the number of CPUs. The square brackets ([ ])indicate that *#num_cpus* is optional. Do not include them in the command.

For example, the following command forces job 123 to run and specifies 1 CPU on hostA and 1 CPU on hostB:

```
brun -m "hostA#1 hostB#1" 123
```

You can only specify a cluster name to forward the job to when the LSF/XL feature is enabled in IBM Spectrum LSF Advanced Edition. In IBM Spectrum LSF Advanced Edition, the **brun -m** command forces a pending job to be forwarded to a remote execution cluster. See *Using IBM Spectrum LSF Advanced Edition* for details.

***job_ID* | "*job_ID*[*index_list*]"**
Required. Specify the job to run, or specify one element of a job array.

-h
Prints command usage to stderr and exits.

-V
Prints LSF release version to stderr and exits.

Limitations

You cannot force a job in SSUSP or USUSP state.

The **brun** command does not guarantee a job runs; it just forces LSF to dispatch the job.

Chapter 55. bsla

Displays information about service classes. Service classes are used in guaranteed resource policies and service-level agreement (SLA) scheduling.

Synopsis

`bsla` [*service_class_name*]

`bsla` [`-h`] [`-V`] [`-N`]

Description

The **`bsla`** command displays the properties of service classes that are configured in the `lsb.serviceclasses` file and dynamic information about the state of each configured service class.

If a default system service class is configured with the **`ENABLE_DEFAULT_EGO_SLA`** parameter in the `lsb.params` file but no other service classes are explicitly configured in the `lsb.serviceclasses` file, the **`bsla`** displays only information for the default SLA.

Options

service_class_name
The name of a service class that is configured in the `lsb.serviceclasses` file.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

-N
Displays information about service class job counts by job slots instead of number of jobs. NSLOTS, and jobs in PEND, RUN, SSUSP, or USUSP state are all counted in slots rather than number of jobs.

Time-based SLA service class output

Time-based SLAs typically have throughput, velocity, or deadline goals. A list of service classes is displayed with the following fields:

SERVICE CLASS NAME
The name of the service class, followed by its description, if any.

PRIORITY
The service class priority. A higher value indicates a higher priority, relative to other service classes. Similar to queue priority, service classes access the cluster resources in priority order.

USER GROUP
User names or user groups who can submit jobs to the service class.

GOAL
The type of service class goal and its configured value. The following values are displayed:

- `THROUGHPUT`
- `VELOCITY`
- `DEADLINE`

ACTIVE WINDOW
The configured time window when the service class goal is active. If no time window is configured for a throughput or velocity goal, the `ACTIVE WINDOW` is `Always Open`.

STATUS
Status of the service class goal.

- `Active:On time` means that the goal is active and meeting its target.
- `Active:Delayed` means that the goal is active but is missing its target.
- `Inactive` means that the goal is not active, and that its time window is closed. Jobs are scheduled as if no service class is defined. LSF does not enforce any service-level goal for an inactive SLA.

THROUGHPUT
For throughput goals, the configured job throughput (finished jobs per hour) for the service class.

SLA THROUGHPUT
The current throughput for the SLA finished jobs per clean period.

ESTIMATED FINISH TIME
For goals with a time window, estimated finish time of the SLA. If the service class status is on time, the finish time is before the configured deadline. If the service class status is delayed, the service class is missing its goal and the **bsla** command shows a finish time later than the deadline.

OPTIMUM NUMBER OF RUNNING JOBS
For goals with a time window, the optimum number of jobs that must be running in the service class for the SLA to meet its goal.

NJOBS

The current number of jobs in the specified service class. A parallel job is counted as one job, regardless of the number of job slots it uses.

PEND
The number of pending jobs in the specified service class.

RUN
The number of running jobs in the specified service class.

SSUSP
The number of system-suspended jobs in the service class.

USUSP
The number of user-suspended jobs in the specified service class.

FINISH
The number of jobs in the specified service class in `EXIT` or `DONE` state.

Resource-based SLA service class output

Resource-based SLAs have guarantee goals. A list of service classes is displayed with the following fields:

SERVICE CLASS NAME
The name of the service class, followed by its description, if any.

GOAL
The type of service class goal and its configured value (`GUARANTEE`).

AUTO_ATTACH
Automatic attachment configuration (`Y` or `N`).

ACCESS_CONTROL
Configured access restrictions for the guarantee SLA, if any.

POOL NAME
Name of the guaranteed resource pool.

TYPE
Guaranteed resource type.

GUAR CONFIG
Number of resources in the pool that is guaranteed to the SLA.

GUAR USED
Number of resources within the guarantee in use by the SLA. Resource use includes both running and suspended jobs.

TOTAL USED
Number of resources in the pool currently in use by the SLA. This total can exceed the number of guaranteed resources for the SLA if other guaranteed SLAs that use the same resource pool are not running at capacity. Resource use includes both running and suspended jobs.

USED GUARANTEE HOSTS
Information on the hosts that are allocated from each guarantee pool for the SLA, organized by guarantee pool. This section only displays if there are jobs running in the SLA.

EGO-enabled SLA service class output

In addition to the general output, EGO-enabled SLA service classes display the following fields:

CONSUMER
The name of the EGO consumer from which hosts are allocated to the SLA.

EGO_RES_REQ

The EGO resource requirement that is defined in the SLA.

MAX_HOST_IDLE_TIME
How long the SLA holds its idle hosts before LSF releases them to EGO.

NUM_RECALLED_HOSTS

The number of hosts that are allocated to the SLA that EGO reclaimed.

RECALLED_HOSTS_TIMEOUT

The amount of time EGO gives to LSF to clean up its workload before EGO reclaims the host.

Examples

The following time-based service class that is named `Duncan` is configured in the `lsb.serviceclasses` file:

```
Begin ServiceClass
NAME = Duncan
CONSUMER = Duncan
PRIORITY = 23
USER_GROUP = user1 user2
GOALS = [VELOCITY 8 timeWindow (9:00-17:30)] \
[DEADLINE timeWindow (17:30-9:00)]
DESCRIPTION = Daytime/Nighttime SLA
End ServiceClass
```

The **bsla** command shows the following properties and status:

```
bsla Duncan
SERVICE CLASS NAME:  Duncan
 -- Daytime/Nighttime SLA
PRIORITY:  23
CONSUMER: Duncan
EGO_RES_REQ: any host
MAX_HOST_IDLE_TIME: 120
USER_GROUP:  user1 user2

GOAL:  VELOCITY 8
ACTIVE WINDOW: (9:00-17:30)
STATUS:  Active:On time
SLA THROUGHPUT:  0.00 JOBS/CLEAN_PERIOD
GOAL:  DEADLINE
ACTIVE WINDOW: (17:30-9:00)
STATUS:  Inactive
SLA THROUGHPUT:  0.00 JOBS/CLEAN_PERIOD

 NJOBS   PEND    RUN     SSUSP   USUSP   FINISH
      0      0       0        0       0       0
```

The following resource pools that are named `linuxPool` and `solarisPool` are configured in the `lsb.resources` file:

```
Begin GuaranteedResourcePool
NAME =linuxPool
TYPE = hosts
HOSTS = linuxHG
DISTRIBUTION = [[sla1,10%] [sla2,25%]
DESCRIPTION = A linux resource pool used by sla1, and sla2.
End GuaranteedResourcePool
```

```
Begin GuaranteedResourcePool
NAME =solarisPool
TYPE = hosts
HOSTS = solarisHG
DISTRIBUTION = [[sla1,20%] [sla2,30%] [sla3,25%]]
DESCRIPTION = A solaris resource pool used by sla1, sla2, and sla3.
End GuaranteedResourcePool
```

The **bsla** command shows the following information for service class `sla1`:

```
bsla sla1
SERVICE CLASS NAME:  sla1
 -- SLA ONE
ACCESS CONTROL:    QUEUES[normal]  FAIRSHARE__GROUPS[lsfadmins/]
AUTO ATTACH:       Y
GOAL:              GUARANTEE                    GUARANTEE    GUARANTEE    TOTAL
POOL NAME      TYPE    CONFIG         USED     USEDslotPool       slots       10
0        0
```

See also

bresources, **bhist**, **bjobs**, **bkill**, **bmod**, **bsub**, `lsb.acct`, `lsb.serviceclasses`, `lsb.resources`

Chapter 56. bslots

Displays slots available and backfill windows available for backfill jobs.

Synopsis

`bslots [-l] [-n slots] [-R "res_req"] [-W [hour:]minutes]`

`bslots [-h | -V]`

Description

The available slots that are displayed by the **bslots** command are not used for running jobs and can be used for backfill jobs. The **bslots** command displays a snapshot of the slots not in use by parallel jobs or advance reservations. They might not be available at job submission.

By default, displays all available slots, and the available run times (backfill windows) for those slots. When no slots are available for backfill, the **bslots** command displays the following message:

```
No backfill window exists at this time
```

The **bslots** command calculates the backfill window based on the estimated start time of potential backfill jobs. Estimated start time is only relatively accurate according to current running job information. If running jobs finish earlier or later, estimated start time might be moved to earlier or later time. You might see a small delay of a few minutes between the job finish time on which the estimate was based and the actual start time of the allocated job.

If the available backfill window has no runtime limit, its length is displayed as `UNLIMITED`.

LSF does not calculate predicted start times for pending reserve jobs if no backfill queue is configured in the system. In that case, the resource reservation for pending jobs works as normal, but no predicted start time is calculated, and the **bslots** command does not show the backfill window.

Options

-l

Displays backfill windows in a long multi-line format. The **-l** option displays host names and the number of slots on each host available for backfill.

-n *slots*

Specifies required slots (processors). Backfill windows whose widths are equal or larger than specified value are returned.

When no slots are available for backfill, the **bslots -n** command displays the following message:

```
No backfill window meets these requirements at this time
```

-R "*res_req*"

Selects hosts for calculating the backfill windows according to the specified resource requirement. By default, selects hosts of any type. The `-R` option supports only the `select` resource requirement string. Other resource requirement sections are not supported.

If the **LSF_STRICT_RESREQ=y** is set in the `lsf.conf` file, the selection string must conform to the stricter resource requirement string syntax described in *Administering IBM Spectrum LSF*. The strict resource requirement syntax applies to only the `select` section.

When no slots are available for backfill, the **bslots -R** command displays the following message:

```
No backfill window meets these requirements at this time
```

-W [*hour*:]*minutes*
Specifies expected runtime limit. Backfill windows whose lengths are equal or larger than specified value are returned.

When no slots are available for backfill, the **bslots -W** command displays the following message:

```
No backfill window meets these requirements at this time
```

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Chapter 57. bstage

Stages data files for jobs with data requirements by copying files or creating symbolic links for them between the local staging cache and the job execution environment. You must run **bstage** only within the context of an LSF job (like **blaunch**). To access a file with the **bstage** command, you must have permission to read it.

bstage in

Stages in data files for jobs with data requirements. **bstage** copies or symbolically links files *from* the data manager staging area *to* the job execution host.

Synopsis

`bstage in -all` [`-dst` *path*] [`-link`]

`bstage in -src` "[*host_name*`:`]/*abs_file_path*/*file_name*" [`-dst` *path*[/*file_name*]] [`-link`]

`bstage in -src` "[*host_name*`:`]/*abs_folder_path*/[`*`]" [`-dst` *path*[/*file_name*]] [`-link`]

`bstage in -tag` *tag_name* [`-u` *user_name*] [`-dst` *path*] [`-link`]

Description

Copy or symbolically link files *from* the data manager staging cache *to* the job execution host. You must specify one of the following options: `-all`, `-src`, or `-tag` *tag_name*.

If the containing job is an array element, the **bstage** command checks that a subdirectory exists in the job staging area that corresponds to the array index of the containing job.

By default, the required files are staged into the local staging area cache as soon as the job is submitted. The **bstage in** command inside the job finds the location of the file in the cache. **bstage in** copies (**cp** or **scp**) or links (**ln**) the file from the cache location to the job current working directory.

Options

-all

Copy all the files that are requested with the job submission to the job current working directory. The command finds the location of each requested stage in file in the cache. All files are copied to the folder in a flat directory structure. Input files with the same name overwrite one another.

Essentially this option is a shortcut for the following command:

```
bstage in -src "host_name:/abs_file_path/file_name" -dst path/file_name
```

To copy entire folders but preserve the directory structure, use the `-src` option with a directory wildcard (either / or /*).

When you use the asterisk character (*) at the end of the path, the data requirements string must be in quotation marks.

-dst *path*

The destination folder for the staged files.

The target of the copy can be a relative path to the job current working directory, or an absolute path. If any directories in the path do not exist, **bstage in** attempts to create them. If you do not specify `-dst`, the default is the job execution current working directory.

If the path does not exist and `-src` specifies a single file, the path is interpreted as the destination file to copy to.

If the path exists and `-src` is a single file, the file is either copied or replaced:

- If path is a file, the file is replaced with the new file.
- If path is a directory, the file is copied into the directory under its original name.

If you specify `-tag` or `-all`, or you specify `-src` with a directory wildcard, the destination is interpreted as a folder name relative to the job current working directory. If this directory does not exist, LSF attempts to create it.

-src "[*host_name*:]/*abs_file_path*/*file_name*"

Copy only the file that is requested with the *host_name*:*abs_file_path* option in the job submission to your current working directory. The host and file path specification must match the requirement that was specified when the job was submitted. Use the **bjobs -data** command to see the exact host and file path specification:

```
bjobs -data 1962
JOBID   USER    STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME   SUBMIT_TIME
1962    user1   PEND  normal hostA                *p 1000000 Sep 20 16:31
FILE                                SIZE   MODIFIED
datahost:/proj/user1/input1.dat     500 M   Jun 27 16:37:52
datahost:/proj/user1/input2.dat     100 M   Jun 27 16:37:52
datahost:/proj/user1/input3.dat     -      -
```

You can omit the host name to request files locally accessible on the submission host.

-src "[*host_name*:]/*abs_folder_path*/[*]"

Copy the contents of the folder that is requested with the *host_name*:/*abs_folder_path*/ option in the job submission to your current working directory. The host and file path specification must match the requirement that was specified when the job was submitted.

You can omit the host name to request files locally accessible on the submission host.

If you specify a folder name without a file name, the absolute path must terminate in a directory (/*) or recursive directory (/) wildcard character. In this case, the `-dst` option is interpreted as a folder, and all files are downloaded to the appropriate subdirectories, replicating the underlying structure.

When you use the asterisk character (*) at the end of the path, the data requirements string must be in quotation marks.

For example, the following job has a data requirement that requests a recursive directory:

```
bsub -data “hostA:/tmp/” ...
```

LSF stages the entire /tmp directory and all subdirectories on hostA. Your job can then call that directory with the **bstage in** command:

```
bstage in -src "hostA:/tmp/" -dst directory
```

LSF replicates the entire subdirectory structure under *directory* in the job execution current working directory.

-link

Create symbolic links to the requested source files from the staging area cache location instead of copying them. Use the `-link` option to avoid unnecessary file copying between the execution host and the staging area. The staging area must be directly mounted on the job execution host to create the link.

-tag *tag_name*

Copy all files in the local cache that are associated with the specified tag name to the folder specified by the destination option (`-dst`). If the `-dst` option is specified, the destination is interpreted as a folder, and the entire directory structure under this tag folder is replicated in the destination.

Use the `-tag` option when a job uses an intermediate data file that is created by an earlier job. You must have read permission on the tag directory.

Valid tag names can contain only alphanumeric characters (`[A-z|a-z|0-9]`), and a period (`.`), underscore (_), and dash (`-`). The tag name cannot contain the special operating system names for parent directory (`../`), current directory (`./`), or user home directory (`~/`). Tag names cannot contain spaces. Tag names cannot begin with a dash (`-`).

Use the **bdata tags clean** command to remove tags.

Important: You are responsible for the name space of your tags. LSF does not check whether a tag is valid. Use strings like the job ID, array index, and cluster name as part of your tag names to make sure that your tag is unique.

-u *user_name*

By default, your job can stage in files that are associated only with your own tags. Use the `-u` option to stage in files that are associated with tags that belong to another user name. The **CACHE_ACCESS_CONTROL = Y** parameter must be configured in the `lsf.datamanager` file to use the `-u` option.

You must make sure that the tag exists and that you have appropriate permission to use the files associated with that tag before you submit your job.

bstage out

Stages out data files for jobs with data requirements. The **bstage** command copies or creates symbolic links to files *from* the job current working directory *to* the data management cache, then requests a transfer job to copy the file or folder to a host.

Synopsis

`bstage out -src` *file_path*`/`*file_name* [`-dst` [*host_name*`:`]*path*[`/`*file_name*]] [`-link`]

`bstage out -src` *folder_path*`/` [`-dst` [*host_name*`:`]*path*[`/`*file_name*]] [`-link`]

`bstage out -src` *file_path*`/`*file_name* `-tag` *tag_name* [`-link` | `-g` *user_group_name*]

Description

Copy or symbolically link a file or folder *from* the job current working directory *to* the data manager staging cache.

The **bstage out** command uses the value of the **LSB_DATA_CACHE_TOP** environment variable to find the staging area.

By default (if you specify the `-src` option, but not the `-tag` option), a transfer job is submitted by the LSF data manager to stage the file out from the staging area to the remote destination specified in the `-dst` option. If you specify the `-tag` option, the file or folder is copied only to the tag folder in the staging area and no transfer job is submitted.

Note: With the `-src` option, the transfer job that is submitted to LSF runs asynchronously even after the command returns. When you are staging out different files or folders, use a different destination for each one since the order in which the files are transferred is not guaranteed.

Options

-src *file_path*/*file_name*

Required. Path to the file to be copied from the job execution environment. Relative paths are resolved relative to the job current working directory.

If the `-tag` option is not specified, the path must be to a file. LSF contacts the LSF data manager to determine whether the file exists in the `STAGING_AREA/stgout` directory for the job. If the file does not exist, the file is first copied to the job `stgout` directory in the cache. LSF data manager then contacts LSF to submit a transfer job to LSF to stage the file out to the destination.

If the path contains symbolic links, LSF data manager uses the symbolic link name and copies the contents of the file to the staging area. For example, if the path is `/tmp/linkdir1/outfile1`, and `linkdir1` is a symbolic link to `/home/user1`, the contents of the file `outfile1` are copied to the cache in the appropriate stage out or tag folder under the relative path `tmp/linkdir1/outfile1/`, not `tmp/home/user1/outfile1/`.

-src *folder_path/*

Required. Path to the folder to be copied from the job execution environment. Relative paths are resolved relative to the job current working directory.

The asterisk (*) wildcard character `-src` *`folder_path/*`* is not supported for the **bstage out** command.

LSF contacts the LSF data manager to determine whether the folder exists in the `STAGING_AREA/stgout` directory for the job. If the folder does not exist, the folder is first copied to the job `stgout` directory in the cache. LSF data manager then contacts LSF to submit a transfer job to LSF to stage the folder out to the destination.

If the path contains symbolic links, LSF data manager uses the symbolic link name and copies the contents of the folder to the staging area. For example, if the path is `/tmp/linkdir1/`, and `linkdir1` is a symbolic link to `/home/user1`, the contents of the folder `linkdir1` are copied to the cache in the appropriate stage out or tag folder under the relative path `tmp/linkdir1/`, not `tmp/home/user1/outfile1/`.

-dst [*host_name*:]*path*][/*file_name*]

Path to the final destination of the transfer job that copies the file out of the staging area. If you do not specify the `-dst` option, the submission host and directory that is specified by the **LSB_OUTDIR** environment variable is assumed to be the root and the path that is provided to the `-src` option is appended to this root. The default *host_name* is the submission host.

The following table shows the mapping of the `-dst` argument and ultimate destination sent to the transfer tool command to stage out the job:

Command	Transfer job destination
`-dst` not specified	`$LSB_SUB_HOST:$LSB_OUTDIR`
`-dst` *relative_path*	`$LSB_SUB_HOST:$LSB_OUTDIR/`*`relative_path`*
`-dst` *absolute_path*	`$LSB_SUB_HOST:`*`absolute_path`*
`-dst` *host_name*`:`*absolute_path*	*host_name*`:`*absolute_path*

The argument to the `-dst` option accepts both relative and absolute paths, both with and without host names, but all of these arguments are converted into an absolute `host_name:path` pair according to the table. You cannot use path descriptors that contain special names `~/`, `./`, and `../`.

If the folders in the destination location do not exist, the success or failure of the transfer job depends on the transfer tool configured. For example, the default tool (the **scp** command) does not create destination folders, but other tools (such as the **rsync** command) do.

-link

Create symbolic links from the requested source files to the staging area cache location instead of copying them. Use the `-link` option to avoid unnecessary file copying between the execution host and the staging area. The staging area must be directly mounted on the job execution host to create the link.

Important: You must ensure that the source file will not be cleaned up after the end of the job to avoid the symbolic link from becoming stale before the file is staged out or used by a subsequent job.

-tag *tag_name*

Copy the file to the staging area tag directory associated with *tag_name*. LSF creates the directory if necessary. The LSF data manager does not submit a job to transfer out the file, or create a record for it in the cache. You cannot use the `-tag` option with the `-dst` or `-link` option.

LSF data manager associates the required files to an arbitrary name you choose, and the LSF data manager reports the existence of that tag if you query it with the **bdata tags** command.

Valid tag names can contain only alphanumeric characters (`[A-z|a-z|0-9]`), and a period (`.`), underscore (_), and dash (`-`). The tag name cannot contain the special operating system names for parent directory (`../`), current directory (`./`), or user home directory (`~/`). Tag names cannot contain spaces. Tag names cannot begin with a dash (`-`).

Use the **bdata tags clean** command to remove tags.

You must be the owner of the tag folder to copy files into it.

Important: You are responsible for the name space of your tags. LSF does not check whether the tag is valid. Use strings like the job ID, array index, and cluster name as part of your tag names to make sure that your tag is unique.

-g *user_group_name*

By default, when a job stages out files to a tag, the tag directory is only accessible by the user who submitted the job. When the **CACHE_ACCESS_CONTROL = Y** parameter is configured in the `lsf.datamanager` file, the –g option changes the group that is associated with the tag. The *user_group_name*argument specifies the user group name to be associated with the tag. The permissions on the tag directory and its contents are set so that the specified group can access the files. You can also use the following command to change the group that is associated with a tag:

```
bdata chgrp -g group_name -tag tag_name
```

Help and version options

IBM Spectrum LSF Data Manager help and version display options

`bstage [-h[elp] | -V]`

-h[help]
Displays the command usage of the **bstage** command to `stderr` and exits.

-V
Prints IBM Spectrum LSF Data Manager release version to `stderr` and exits.

See also

bdata, **bhist**, **bjobs**, **bmod**, **bsub**, `lsf.conf`, `lsf.datamanager`

Chapter 58. bstatus

Gets current external job status or sets new job status.

Synopsis

`bstatus [-d "description"] job_ID | "job_ID[index]" | -J job_name`

`bstatus [-h | -V]`

Description

Gets and displays the message description text of a job, or changes the contents of the message description text with the `-d` option. Always operates on the message with index 0.

You can set the external status of a job until it completes. You cannot change the status of done or exited jobs. You can display the status of a job until it is cleaned from the system.

If you specify a job ID, the **bstatus** command has the following behavior:

- You can get the external job status of jobs submitted by other users, but you cannot set job status of jobs that are submitted by other users.
- You can set external status only on your own jobs.
- Only root and LSF administrators can set external job status on jobs that are submitted by other users.

Job names are not unique; if you specify **-J** *job_name* the **bstatus** command has the following behavior:

- You can get or set the external status only on your own jobs.
- You cannot get or set external job status on jobs that are submitted by other users.
- Root and the LSF administrators can get or set the external status only on their own jobs.

Options

-d "*description*"
Updates the job status with specified message description text.

job_ID* | "*job_ID*[*index*]" | -J *job_name
Required. Operates on the specified job.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (*) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Examples

```
bstatus 2500
JOBID    FROM    UPDATE_TIME   STATUS
2500     user1   Sep 14 16:54  step 1
```

Displays the message description text of message index 0 of job 2500.

```
bstatus -d "step 2" 2500
```

Changes the message description text of message index 0 of job 2500 to `step 2`.

See also

bpost, **bread**

Chapter 59. bstop

Suspends unfinished jobs.

Synopsis

`bstop` [`-a`] [`-app` *application_profile_name*] [`-g` *job_group_name*] [`-J` *job_name*] [`-m` *host_name* | `-m` *host_group*] [`-q` *queue_name*] [`-sla` *service_class_name*] [`-u` *user_name* | `-u` *user_group* | `-u all`] [`0`] [*job_ID ...* | "*job_ID*[*index*]"] ...

`bstop` [`-h` | `-V`]

Description

By default, the **bstop** command sends the SIGSTOP signal to sequential jobs and the SIGTSTP signal to parallel jobs to suspend them.

You must specify a job ID or the `-g`, `-J`, `-m`, `-u`, or `-q` option. You cannot suspend a job that is already suspended. Specify job ID `0` (zero) to stop multiple jobs.

Only root and LSF administrators can operate on jobs submitted by other users.

Use the **bresume** command to resume suspended jobs.

An administrator can use the **bstop** command on a job stopped by the user (in the state USUSP) to prevent the user from resuming the job.

You can also use the **bkill -s STOP** command to send the suspend signal to a job or use the **bkill -s TSTP** command to suspend one or more parallel jobs. Use the **bkill -s CONT** command to send a resume signal to a job.

If a signal request fails to reach the job execution host, LSF retries the operation later when the host becomes reachable. LSF retries the most recent signal request.

Options

0
Suspends all the jobs that satisfy other options (`-g`, `-m`, `-q`, `-u`, and `-J`).

-a
Suspends all jobs.

-app *application_profile_name*
Suspends only jobs associated with the specified application profile. You must specify an existing application profile.

-g *job_group_name*
Suspends only jobs in the specified job group.

-J *job_name*
Suspends only jobs with the specified name.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (`*`) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-m *host_name* | -m *host_group*
Suspends only jobs dispatched to the specified host or host group.

-q *queue_name*
Suspends only jobs in the specified queue.

-sla *service_class_name*
Suspends jobs belonging to the specified service class.

Use the **bsla** command to display the properties of service classes configured in the `lsb.serviceclasses` file and dynamic information about the state of each configured service class.

-u *user_name* | -u *user_group* | -u all
Suspends only jobs owned by the specified user or user group, or all users if the keyword `all` is specified. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) on a Windows command prompt or a double backslash (*DOMAIN_NAME**user_name*) on a UNIX or Linux command line.

***job_ID* ... | "*job_ID*[*index*]" ...**
Suspends only the specified jobs. Jobs submitted by any user can be specified here without using the `-u` option.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Examples

```
bstop 314
```

Suspends job number 314.

```
bstop -m hostA
```

Suspends the user's last job that was dispatched to host `hostA`.

```
bstop -u jsmith 0
```

Suspends all the jobs submitted by user `jsmith`.

```
bstop -u all
```

Suspends the last submitted job in the LSF system.

```
bstop -u all 0
```

Suspends all jobs for all users in the LSF system.

```
bstop -g /risk_group/consolidate 0
```

Suspends all jobs in the job group `/risk_group/consolidate`.

```
bstop -app fluent 0
```

Suspends all jobs associated with the application profile `fluent`.

See also

bapp, **bgadd**, **bgdel**, **bhosts**, **bjgroup**, **bjobs**, **bkill**, **bparams**, **bqueues**, **bresume**, **bsub**, **mbatchd**, **kill**, **signal**, `lsb.params`

Chapter 60. bsub

Submits a job to LSF by running the specified command and its arguments.

Synopsis

bsub [*options*] *command* [*arguments*]

bsub [*options*] *job_script*| LSB_DOCKER_PLACEHOLDER

bsub -R "*res_req*" [-R "*res_req*" ...] [*options*] *command* [*arguments*]

bsub -pack *job_submission_file*

bsub -h[elp] [all] [description] [*category_name* ...] [-*option_name* ...]

bsub -V

Categories and options

Use the keyword all to display all options and the keyword description to display a detailed description of the **bsub** command. For more details on specific categories and options, specify **bsub -h** with the name of the categories and options.

Categories

List categories for options of the **bsub** command.

Category: io

Specify input/output files and directories: `-cwd`, `-e`, `-eo`, `-f`, `-i`, `-is`, `-o`, `-oo`, `-outdir`, `-tty`.

Category: limit

Specify limits: `-c`, `-C`, `-D`, `-eptl`, `-F`, `-hl`, `-M`, `-p`, `-ptl`, `-S`, `-T`, `-ul`, `-v`, `-W`, `-We`.

Category: notify

Control user notification: `-B`, `-K`, `-N`, `-u`.

Category: pack

Submit job packs (single files containing multiple job requests): `-pack`.

Category: properties

Specify job submission properties: `-app`, `-ar`, `-csm`, `-data`, `-datagrp`, `-E`, `-env`, `-Ep`, `-g`, `-I`, `-Ip`, `-Is`, `-IS`, `-ISp`, `-ISs`, `-IX`, `-J`, `-Jd`, `-jsdl`, `-jsdl_strict`, `-jsm`, `-k`, `-K`, `-L`, `-P`, `-q`, `-Q`, `-r`, `-rn`, `-rnc`, `-s`, `-sla`, `-smt`, `-sp`, `-stage`, `-step_cgroup`, `-wa`, `-wt`, `-XF`, `-Zs`.

Category: resource

Specify resources and resource requirements: `-alloc_flags`, `-clusters`, `-cn_cu`, `-cn_mem`, `-core_isolation`, `-data`, `-datagrp`, `-freq`, `-gpu`, `-hostfile`, `-ln_mem`, `-ln_slots`, `-m`, `-n`, `-nnodes`, `-network`, `-R`, `-U`.

Category: schedule

Control job scheduling and dispatch: `-a`, `-b`, `-clusters`, `-ext`, `-G`, `-H`, `-Lp`, `-mig`, `-t`, `-ti`, `-U`, `-w`, `-x`.

Category: script

Use job scripts: `-Zs`.

Description

You can build a job file one line at a time, or create it from another file, by running **bsub** without specifying a job to submit. When you do this, you start an interactive session in which **bsub** reads command lines from the standard input and submits them as a single batch job. You are prompted with **bsub**> for each line.

Use **bsub -Zs** to spool a job command file to the directory specified by the **JOB_SPOOL_DIR** parameter in `lsb.params`, and use the spooled file as the command file for the job.

Use the **bmod -Zsn** command to modify or remove the command file after the job has been submitted. Removing or modifying the original input file does not affect the submitted job.

Examples

Write a job file one line at a time

For UNIX, the command lines are run as a Bourne shell (`/bin/sh`) script. Only valid Bourne shell command lines are acceptable in the following case:

```
% bsub -q simulation
bsub> cd /work/data/myhomedir bsub> myjob arg1 arg2 ......
bsub> rm myjob.log
```

```
bsub> ^D
Job <1234> submitted to queue <simulation>.
```

For Windows, the command lines are run as a batch file (.BAT). Only valid Windows batch file command lines are acceptable in the following case:

```
C:\> bsub -q simulation
bsub> cd \\server\data\myhomedir
bsub> myjob arg1 arg2 ......
bsub> del myjob.log
bsub> ^Z
Job <1234> submitted to queue <simulation>.
```

Specify job options in a file

In this example, options to run the job are specified in the `options_file` text file.

```
% bsub -q simulation < options_file
Job <1234> submitted to queue <simulation>.
```

On UNIX, `options_file` must be a text file that contains Bourne shell command lines. It cannot be a binary executable file.

On Windows, `options_file` must be a text file containing Windows batch file command lines.

Spool a job command file

Use **bsub -Zs** to spool a job command file to the directory specified by the **JOB_SPOOL_DIR** parameter in `lsb.params`, and use the spooled file as the command file for the job.

Use the **bmod -Zsn** command to modify or remove the command file after the job has been submitted. Removing or modifying the original input file does not affect the submitted job.

Redirect a script to bsub standard input

You can redirect a script to the standard input of the **bsub** command:

```
% bsub < myscript
Job <1234> submitted to queue <test>.
```

In this example, the `myscript` file contains job submission options as well as command lines to execute. When the **bsub** command reads a script from its standard input, it can be modified right after **bsub** returns for the next job submission.

When the script is specified on the **bsub** command line, the script is not spooled:

```
% bsub myscript
Job <1234> submitted to default queue <normal>.
```

In this case, the command line `myscript` is spooled, instead of the contents of the `myscript` file. Later modifications to the `myscript` file can affect job behavior.

Specify embedded submission options

You can specify job submission options in scripts read from standard input by the **bsub** command using lines starting with #BSUB:

```
% bsub -q simulation bsub> #BSUB -q test
bsub> #BSUB -o outfile -R "mem>10"
bsub> myjob arg1 arg2
bsub> #BSUB -J simjob
bsub> ^D
Job <1234> submitted to queue <simulation>.
```

Note:

- Command-line options override embedded options. In this example, the job is submitted to the `simulation` queue rather than the `test` queue.
- Submission options can be specified anywhere in the standard input. In the above example, the `-J` option of **bsub** is specified after the command to be run.
- More than one option can be specified on one line, as shown in the example above.

Run a job under a particular shell

By default, LSF runs batch jobs using the Bourne (`/bin/sh`) shell. You can specify the shell under which a job is to run. This is done by specifying an interpreter in the first line of the script.

For example:

```
% bsub
bsub> #!/bin/csh -f
bsub> set coredump=‘ls |grep core‘
bsub> if ( "$coredump" != "") then
bsub> mv core core.‘date | cut -d" " -f1‘
bsub> endif
bsub> myjob
bsub> ^D
Job <1234> is submitted to default queue <normal>.
```

The **bsub** command must read the job script from standard input to set the execution shell. If you do not specify a shell in the script, the script is run using `/bin/sh`. If the first line of the script starts with a `#` not immediately followed by an exclamation mark (`!`), then `/bin/csh` is used to run the job.

For example:

```
% bsub
bsub> # This is a comment line. This tells the system to use /bin/csh to
bsub> # interpret the script.
bsub>
bsub> setenv DAY ‘date | cut -d" " -f1‘
bsub> myjob bsub> ^D
Job <1234> is submitted to default queue <normal>.
```

If running jobs under a particular shell is required frequently, you can specify an alternate shell using a command-level job starter and run your jobs interactively.

Options

List of options for the **bsub** command.

-a

Specifies one or more application-specific **esub** or **epsub** executable files that you want LSF to associate with the job.

Categories

schedule

Synopsis

`bsub -a "`*application_name* [([*argument*[,*argument*...]])]..."

Description

The value of `-a` must correspond to the application name of an actual **esub** or **epsub** file. For example, to use **bsub -a fluent**, one or both of the **esub.fluent** or **epsub.fluent** files must exist in **LSF_SERVERDIR**.

For example, to submit a job that invokes the application-specific **esub** (or **epsub**) executables named **esub.license** (or **epsub.license**) and **esub.fluent** (or **epsub.fluent**), enter:

```
bsub -a "license fluent" my_job
```

The name of the application-specific **esub** or **epsub** program is passed to the master **esub**. The master **esub** program (LSF_SERVERDIR/**mesub**) handles job submission requirements of the application. Application-specific **esub** programs can specify their own job submission requirements. The value of `-a` is set in the **LSB_SUB_ADDITIONAL** environment variable.

mesub uses the method name license to invoke the **esub** named LSF_SERVERDIR/`esub.license` or the **epsub** named LSF_SERVERDIR/`epsub.license`, and the method name fluent to invoke the **esub** named LSF_SERVERDIR/`esub.fluent` or the **epsub** named LSF_SERVERDIR/`epsub.fluent`. Therefore, **mesub** is run twice: Before the job is submitted to **mbatchd** to run the **esub** scripts, and after the job is submitted to run the **epsub** scripts.

LSF first invokes the executable file named **esub** (without *.application_name* in the file name) if it exists in **LSF_SERVERDIR**, followed by any mandatory **esub** or **epsub** executable files that are defined using the parameter **LSB_ESUB_METHOD** in the `lsf.conf` file, and then any application-specific **esub** executable files (with *.application_name* in the file name) specified by `-a`. After the job is submitted, LSF invokes the executable file named **epsub** (without *.application_name* in the file name) if it exists in **LSF_SERVERDIR**, followed by any mandatory **epsub** executable files (as specified using the parameter **LSB_ESUB_METHOD**), and then any application-specific **epsub** executable files (with *.application_name* in the file name) specified by `-a`.

The name of the **esub** or **epsub** program must be a valid file name. It can contain only alphanumeric characters, underscore (_) and hyphen (-).

Restriction: If the value of `-a` corresponds to a value in **LSB_ESUB_METHOD**, this value must correspond to an actual **esub** or **epsub** file in **LSF_SERVERDIR**. For example, if `fluent` is defined in **LSB_ESUB_METHOD**, the `esub.fluent` or `epsub.fluent` file must exist in **LSF_SERVERDIR** to use **bsub -a fluent**.

If you have an **esub** or **epsub** that runs an interactive or X-window job and you have SSH enabled in `lsf.conf`, the communication between hosts is encrypted.

esub arguments provide flexibility for filtering and modifying job submissions by letting you specify options for **esub** executable files. **epsub** provides the flexibility to perform external logic using information about jobs that are just submitted, such as the assigned job ID and queue name.

The variables you define in the **esub** and **epsub** arguments can include environment variables and command output substitution.

Note: The same arguments that are passed to **esub** are also passed to **epsub**. You cannot pass different arguments to an **esub** file and an **epsub** file with the same application name.

Valid **esub** and **epsub** arguments can contain alphanumeric characters, spaces, special characters (` " \ $ !) and other characters (~@#%^&*( ) -=_+[] | {};':,./<>?). Special patterns like variables (for example, $PATH) and program output (for example, **`ls`** command output) in an **esub** or **epsub** argument will also be processed.

For example, if you use **bsub -a "esub1 ($PATH, `ls`)" user_job**, the first argument passed to **esub1** would be the value of variable *PATH*, and the second argument passed to **esub1** would be the output of command ls.

You can include a special character in an **esub** or **epsub** argument with an escape character or a pair of apostrophes ('). The usage may vary among different shells. You can specify an argument containing separators ('(', ')', ',') and space characters (' ').

You can also use an escape character (\) to specify arguments containing special characters, separators and space characters. For example:

bsub -a "application_name1(var1,var2 contain \(\)\,)" user_job

For fault tolerance, extra space characters are allowed between entities including **esub**/**epsub**, separators, and arguments. For example, the following is valid input:

```
bsub -a " esub1 ( var1 , var2 ) " user_job
```

The maximum length allowed for an **esub**/**epsub** argument is 1024 characters. The maximum number of arguments allowed for an **esub**/**epsub** is 128.

Jobs submitted with an **esub** (or **epsub**) will show all the **esubs** in **bjobs -l** output, first with the default and then user **esubs**.

Examples

- To specify a single argument for a single **esub**/**epsub** executable file, use:

  ```
  bsub –a “application_name(var1)” user_job
  ```
- To specify multiple arguments for a single **esub**/**epsub** executable file, use:

  ```
  bsub –a “application_name(var1,var2,...,varN)” user_job
  ```
- To specify multiple arguments including a string argument for a single **esub**/**epsub** executable file, use:

  ```
  bsub –a “application_name(var1,var2 is a string,...,varN)” user_job
  ```
- To specify arguments for multiple **esub**/**epsub** executable files, use:

  ```
  bsub –a “application_name1(var1,var2) application_name2(var1,var2)” user_job
  ```
- To specify no argument to an **esub**/**epsub** executable file, use:

  ```
  bsub –a “application_name1” user_job
  ```

-alloc_flags

Specifies the user level CSM allocation prologs and epilogs.

Categories

resource

Synopsis

`bsub -alloc_flags "`*flag1* [*flag2* ...]"

Description

This option is for LSF jobs that are submitted with the IBM Cluster Systems Manager (CSM) integration package.

Specify an alphanumeric string of flags and separate multiple flags with a space.

-app

Submits the job to the specified application profile.

Categories

properties

Synopsis

`bsub -app` *application_profile_name*

Description

You must specify an existing application profile. If the application profile does not exist in `lsb.applications`, the job is rejected.

-ar

Specifies that the job is autoresizable.

Categories

properties

Synopsis

```
bsub -ar
```

-B

Sends mail to you when the job is dispatched and begins execution.

Categories

notify

Synopsis

```
bsub -B
```

-b

Dispatches the job for execution on or after the specified date and time.

Categories

schedule

Synopsis

bsub -b [[*year*:][*month*:]*day*:]*hour*:*minute*

Description

The date and time are in the form of [[*year*:][*month*:]*day*:]*hour*:*minute* where the number ranges are as follows: year after 1970, month 1-12, day 1-31, hour 0-23, minute 0-59.

At least two fields must be specified. These fields are assumed to be hour:minute. If three fields are given, they are assumed to be day:hour:minute, four fields are assumed to be *month*:*day*:*hour*:*minute*, and five fields are assumed to be *year*:*month*:*day*:*hour*:*minute*.

If the year field is specified and the specified time is in the past, the start time condition is considered reached and LSF dispatches the job if slots are available.

Examples

```
bsub -b 20:00 -J my_job_name my_program
```

Submit `my_program` to run after 8 p.m. and assign it the job name `my_job_name`.

-C

Sets a per-process (soft) core file size limit for all the processes that belong to this job.

Categories

limit

Synopsis

bsub -C *core_limit*

Description

For more information, see **getrlimit(2)**.

By default, the limit is specified in KB. Use **LSF_UNIT_FOR_LIMITS** in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

The behavior of this option depends on platform-specific UNIX or Linux systems.

In some cases, the process is sent a SIGXFSZ signal if the job attempts to create a core file larger than the specified limit. The SIGXFSZ signal normally terminates the process.

In other cases, the writing of the core file terminates at the specified limit.

-c

Limits the total CPU time the job can use.

Categories

limit

Synopsis

`bsub -c` [*hour*`:`]*minute*[*/host_name* | */host_model*]

Description

This option is useful for preventing runaway jobs or jobs that use up too many resources. When the total CPU time for the whole job has reached the limit, a SIGXCPU signal is first sent to the job, then SIGINT, SIGTERM, and SIGKILL.

If **LSB_JOB_CPULIMIT** in `lsf.conf` is set to n, LSF-enforced CPU limit is disabled and LSF passes the limit to the operating system. When one process in the job exceeds the CPU limit, the limit is enforced by the operating system.

The CPU limit is in the form of [*hour*`:`]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The CPU time you specify is the *normalized* CPU time. This is done so that the job does approximately the same amount of processing for a given CPU limit, even if it is sent to host with a faster or slower CPU. Whenever a normalized CPU time is given, the actual time on the execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert a slash (/) between the CPU limit and the host name or model name. If a host name or model name is not given, LSF uses the default CPU time normalization host defined at the queue level (**DEFAULT_HOST_SPEC** in `lsb.queues`) if it has been configured, otherwise uses the default CPU time normalization host defined at the cluster level (**DEFAULT_HOST_SPEC** in `lsb.params`) if it has been configured, otherwise uses the submission host.

Jobs submitted to a chunk job queue are not chunked if the CPU limit is greater than 30 minutes.

Examples

```
bsub -q "queue1 queue2 queue3" -c 5 my_program
```

Submit `my_program` to one of the candidate queues: `queue1`, `queue2`, and `queue3` that are selected according to the CPU time limit specified by **`-c 5`**.

-clusters

LSF multicluster capability only. Specifies cluster names when submitting jobs.

Categories

resource, schedule

Synopsis

`bsub -clusters "all [~cluster_name] ... | cluster_name[+[pref_level]] ... [others[+[pref_level]]]"`

Description

You can specify cluster names when submitting jobs for LSF multicluster capability.

The **-clusters** option has the following keywords:

all
Specifies both local cluster and all remote clusters in the **SNDJOBS_TO** parameter of the target queue in **lsb.queues**. For example:

```
bsub -clusters all -q <send_queue>
```

LSF will go through the **SNDJOBS_TO** parameter in `lsb.queues` to check whether asked clusters (except for the local cluster) are members of **SNDJOBS_TO**. If any cluster except the local cluster does not exist in **SNDJOBS_TO**, the job is rejected with an error message.

others
Sends the job to all clusters except for the clusters you specify. For example:

```
bsub -clusters "c1+3 c2+1 others+2"
```

~
Must be used with all to indicate the rest of the clusters, excluding the specified clusters.

+
When followed by a positive integer, specifies job level preference for requested clusters. For example:

```
bsub -clusters "c1+2 c2+1"
```

If the local cluster name is local_c1, and **SNDJOBS_TO**=`q1@rmt_c1 q2@rmt_c2 q3@rmt_c3`, then the requested cluster should be `local_c1` and `rmt_c3`. For example:

```
bsub -clusters "all ~rmt_c1 ~rmt_c2"
```

-clusters local_cluster restricts the job for dispatch to local hosts. To run a job on remote clusters only, use:

```
bsub -clusters "all ~local_cluster"
```

A job that only specifies remote clusters will not run on local hosts. Similarly, a job that only specifies local clusters will not run on remote hosts. If a job specifies local and remote clusters, the job tries local hosts first, then remote clusters.

If there are multiple default queues, then when `bsub -clusters remote_clusters` is issued, the job is sent to the queue whose **SNDJOBS_TO** contains the requested clusters. For example:

```
bsub -clusters "c2" , DEFAULT_QUEUE=q1 q2, q1: SNDJOBS_TO=recvQ1@c1 recvQ2@c3,
q2: SNDJOBS_TO=recvQ1@c1 recvQ2@c2
```

The job is sent to q2.

To have the job try to run on only local hosts:

```
bsub -q mc -clusters local_c1
```

To have the job try to run on only remote hosts (e.g., `rmt_c1` and `rmt_c2`):

```
bsub -q mc -clusters rmt_c1 rmt_c2
```

To have the job first try local hosts, then `rmt_c1` and `rmt_c2`:

```
bsub -q mc -clusters local_c1 rmt_c1 rmt_c2
```

To ignore preference for the local cluster (because the local cluster is always tried first, even though remote clusters have higher job level preference) and try remote clusters:

```
bsub -q mc -clusters local_c1 rmt_c1+2 rmt_c2+1
```

To have the job first try the local cluster, then remote clusters:

```
bsub -q mc -clusters all
```

To have the job first try the local cluster, then try all remote clusters except for `rmt_c1`:

```
bsub -q mc -clusters all ~rmt_c1
```

To have the job try only remote clusters:

```
bsub -q mc -clusters all ~local_c1
```

The `-clusters` option is supported in esub, so LSF administrators can direct jobs to specific clusters if they want to implement flow control. The `-m` option and `-clusters` option cannot be used together.

-cn_cu

Specifies the cu (compute unit) resource requirement string for the compute node for the CSM job.

Categories

resource

Synopsis

`bsub -cn_cu` *cu_string*

Conflicting options

Do not use with the `-csm y` option.

Description

This option is for easy mode LSF job submission to run with the IBM Cluster Systems Manager (CSM) integration package.

-cn_mem

Specifies the memory limit that is required on the compute node for the CSM job.

Categories

resource

Synopsis

`bsub -cn_mem` *mem_size*

Conflicting options

Use only with the `-csm y` option.

Description

This option is for expert mode LSF job submission to run with the IBM Cluster Systems Manager (CSM) integration package.

-core_isolation

Disables or enables core isolation.

Categories

resource

Synopsis

```
bsub -core_isolation 0|1|2|3|4|5
```

Description

This option is for LSF jobs that are submitted with the IBM Cluster Systems Manager (CSM) integration package.

Specify 0 to disable core isolation.

Specify any integer between 1 and 5 to enable core isolation. This number is passed to CSM.

Default

0. Core isolation is disabled.

-csm

Enables expert mode for the CSM job submission, which allows you to specify a full resource requirement string for CSM resources.

Categories

properties

Synopsis

```
bsub -csm y|n
```

Description

This option is for LSF jobs that are submitted with the IBM Cluster Systems Manager (CSM) integration package.

Default

n. LSF uses easy mode when submitting jobs with CSM options.

-cwd

Specifies the current working directory for job execution.

Categories

io

Synopsis

```
bsub -cwd "current_working_directory"
```

Description

The system creates the current working directory (CWD) if the path for the CWD includes dynamic patterns for both absolute and relative paths. LSF cleans the created CWD based on the time to live value set in the **JOB_CWD_TTL** parameter of the application profile or in `lsb.params`.

The path can include the following dynamic patterns, which are case sensitive:

- %J - job ID
- %JG - job group (if not specified, it will be ignored)
- %I - index (default value is 0)
- %EJ - execution job ID
- %EI - execution index
- %P - project name
- %U - user name
- %G - user group
- %H - first execution host name

If the job is submitted with `-app` but without `-cwd`, and **LSB_JOB_CWD** is not defined, then the application profile defined **JOB_CWD** will be used. If **JOB_CWD** is not defined in the application profile, then the **DEFAULT_JOB_CWD** value is used.

In forwarding mode, if a job is not submitted with the `-cwd` option and **LSB_JOB_CWD** is not defined, then **JOB_CWD** in the application profile or the **DEFAULT_JOB_CWD** value for the execution cluster is used.

LSF does not allow environment variables to contain other environment variables to be expanded on the execution side.

By default, if the current working directory is not accessible on the execution host, the job runs in `/tmp` (on UNIX) or `c:\LSF`*`version_num`*`\tmp` (on Windows). If the environment variable **LSB_EXIT_IF_CWD_NOTEXIST** is set to Y and the current working directory is not accessible on the execution host, the job exits with the exit code 2.

Examples

The following command creates `/scratch/jobcwd/user1/<jobid>_0/` for the job CWD:

```
bsub -cwd "/scratch/jobcwd/%U/%J_%I" myjob
```

The system creates `submission_dir/user1/<jobid>_0/` for the job's CWD with the following command:

```
bsub -cwd "%U/%J_%I" myprog
```

If the cluster wide CWD was defined and there is no default application profile CWD defined:

DEFAULT_JOB_CWD =/scratch/jobcwd/ %U/%J_%I

then the system creates: `/scratch/jobcwd/user1/<jobid>_0/` for the job's CWD.

-D

Sets a per-process (soft) data segment size limit for each of the processes that belong to the job.

Categories

limit

Synopsis

`bsub -D` *data_limit*

Description

For more information, see **getrlimit**(2). The limit is specified in KB.

This option affects calls to `sbrk()` and `brk()` . An `sbrk()` or `malloc()` call to extend the data segment beyond the data limit returns an error.

Note: Linux does not use `sbrk()` and `brk()` within its `calloc()` and `malloc()`. Instead, it uses (`mmap()`) to create memory. DATALIMIT cannot be enforced on Linux applications that call `sbrk()` and `malloc()`.

-data

Specifies data requirements for a job.

Categories

properties, resource

Synopsis

`-data "[`*host_name*`:]`*abs_file_path* `[[`*host_name*`:/]`*abs_file_path* `...]" ... [-datagrp "`*user_group_name*`"] [-datachk] | -data "tag:[`*tag_owner*`@]`*tag_name* `[tag:[`*tag_owner*`@]`*tag_name* `...]" ...`

`-data "[`*host_name*`:]`*abs_folder_path*`[/[*]] [[`*host_name*`:/]`*abs_folder_path*`/[*] ...]" ... [-datagrp "`*user_group_name*`"] [-datachk] | -data "tag:[`*tag_owner*`@]`*tag_name* `[tag:[`*tag_owner*`@]`*tag_name* `...]" ...`

Description

You can specify data requirements for the job three ways:

- As a list of files for staging.
- An absolute path to a folder that contains files for staging.
- As a list of arbitrary tag names.

Your job can specify multiple `-data` options, but all the requested data requirements must be either tags, files, or folders. You can specify individual files, folders, or data specification files in each `-data` clause. You cannot mix tag names with file specifications in the same submission. The combined requirement of all `-data` clauses, including the requirements that are specified inside specification files, are combined into a single space-separated string of file requirements.

Valid file or tag names can contain only alphanumeric characters (`[A-z|a-z|0-9]`), and a period (`.`), underscore (_), and dash (-). The tag or file name cannot contain the special operating system names for parent directory (`../`), current directory (`./`), or user home directory (~/). File, folder, and tag names cannot contain spaces. Tag names cannot begin with a dash character.

A list of files must contain absolute paths to the required source files. The list of files can refer to actual data files your job requires or to a data specification file that contains a list of the data files. The first line of a data specification file must be `#@dataspec`.

A data requirement that is a folder must contain an absolute path to the folder. The colon character (`:`) is supported in the path of a job requirement.

A host name is optional. The default host name is the submission host. The host name is the host that LSF attempts to stage the required data file from. You cannot substitute IP addresses for host names.

By default, if the **CACHE_PERMISSIONS=group** parameter is specified in `lsf.datamanager`, the data manager uses the primary group of the user who submitted the job to control access to the cached files. Only users that belong to that group can access the files in the data requirement. Use the `-datagrp` option to specify a particular user group for access to the cache. The user who submits the job must be a member of the specified group. You can specify only one `-datagrp` option per job.

The *tag_name* element is any string that can be used as a valid data tag.

If the **CACHE_ACCESS_CONTROL = Y** parameter is configured in the `lsf.datamanager` file, you can optionally specify a user name for the tag owner `tag:[`*tag_owner*`@]`*tag_name*.

The sanity check for the existence of files or folders and whether the user can access them, discovery of the size and modification of the files or folders, and generation of the hash from the **bsub** command

occurs in the transfer job. This equalizes submission performance between jobs with and without data requirements. The -datachk option forces the **bsub** and **bmod** commands to perform these operations. If the data requirement is for a tag, the -datachk option has no effect.

You can use the %I runtime variable in file paths. The variable resolves to an array index in a job array submission. If the job is not an array, %I in the data specification is interpreted as 0. You cannot use the %I variable in tag names.

The path to the data requirement files or the files that are listed inside data specification files can contain wildcard characters. All paths that are resolved from wildcard characters are interpreted as files to transfer. The path to a data specification file cannot contain wildcard characters.

The use of wildcard characters has the following rules:

- A path that ends in a slash followed by an asterisk (/*) is interpreted as a directory. For example, a path like /data/august/*. Transfers only the files in the august directory without recursion into subdirectories.
- A path that ends in a slash (/) means that you want to transfer all of the files in the folder and all files in all of its sub folders, recursively.
- You can use the asterisk (*) wildcard character only at the end of the path. For example, a path like /data/*/file_*.* is not supported. The following path is correct: /data/august/*. The * is subject to expansion by the shell and must be quoted properly.
- When wildcard characters are expanded, symbolic links in a path are expanded. If a broken symbolic link is detected, the job submission fails.
- When you use the asterisk character at the end of the path, the data file requirements must be in quotation marks.

Examples

The following job requests three data files for staging, listing in the -data option:

```
bsub -o %J.out -data "hostA:/proj/std/model.tar
hostA:/proj/user1/case02341.dat hostB:/data/human/seq342138.dna" /share/bin/job_copy.sh
 Job <1962> is submitted to default queue <normal>.
```

The following job requests the same data files for staging. Instead of listing them individually in the -data option, the required files are listed in a data specification file named /tmp/dataspec.user1:

```
bsub -o %J.out -data “/tmp/dataspec.user1” /share/bin/job_copy.sh
 Job <1963> is submitted to default queue <normal>.
```

The data specification file /tmp/dataspec.user1 contains the paths to the required files:

```
cat /tmp/dataspec.user1
 #@dataspec
 hostA:/proj/std/model.tar
 hostA:/proj/user1/case02341.dat
 hostB:/data/human/seq342138.dna
```

The following job requests all data files in the folder /proj/std/on hostA. Only the contents of the top level of the folder are requested.

```
bsub -o %J.out -data "hostA:/proj/std/*" /share/bin/job_copy.sh
 Job <1964> is submitted to default queue <normal>.
```

The following job requests all data files in the folder /proj/std/, including on hostA. All files in all subfolders are requested recursively as the required data files.

```
bsub -o %J.out -data "hostA:/proj/std/" /share/bin/job_copy.sh
 Job <1964> is submitted to default queue <normal>.
```

The following command submits an array job that requests data files by array index. All data files must exist.

```
bsub -o %J.out -J "A[1-10]" -data "hostA:/proj/std/input_%I.dat"
/share/bin/job_copy.sh
 Job <1965> is submitted to default queue <normal>.
```

The following job requests data files by tag name.

```
bsub -o %J.out -J "A[1-10]" -data "tag:SEQ_DATA_READY" "tag:SEQ_DATA2"
/share/bin/job_copy.sh
 Job <1966> is submitted to default queue <normal>.
```

The following job requests the data file `/proj/std/model.tar`, which belongs to the user group `design1`:

```
bsub -o %J.out -data "hostA:/proj/std/model.tar" -datagrp "design1" my_job.sh
 Job <1962> is submitted to default queue <normal>.
```

-datachk

For jobs with a data requirement, enables the **bsub** and **bmod** commands to perform a full sanity check on the files and folders, and to generate the hash for each file and folder.

Categories

properties, resource

Synopsis

`-data [-datachk]`

Description

For jobs with a data requirement, this command option forces the **bsub** and **bmod** commands to perform the following operations on the files or folders:

- Sanity check to see if the files or folders exist
- Check whether the submission user can access the files or folders
- Discover the size and modification times of the files or folders
- Generate a hash for each file or folder

If you do not specify this command option, these operations occur at the transfer job by default. You can also force the **bsub** and **bmod** commands to perform these operations by setting the `LSF_DATA_BSUB_CHKSUM` parameter to y or Y in the `lsf.conf` file.

This option only works for jobs with a data requirement. If the data requirement is for a tag, this option has no effect.

-datagrp

Specifies user group for job data requirements.

Categories

properties, resource

Synopsis

`[-datagrp "`*user_group_name* `"]`

Description

By default, if the **CACHE_PERMISSIONS=group** parameter is specified in `lsf.datamanager`, the data manager uses the primary group of the user who submitted the job to control access to the cached files. Only users who belong to that group can access the files in the data requirement. Use the `-datagrp`

option to specify a particular user group for access to the cache. The user who submits the job must be a member of the specified group. You can specify only one `-datagrp` option per job.

The job must have a data requirement to use `-datagrp`.

Normally, if you submit a data job with the `-datagrp` option, you must belong to the data group specified. If you do not belong to the specified group, the **mbatchd** daemon rejects the job submission. If the **LSF_DATA_SKIP_GROUP_CHECK=Y** parameter is configured in the `lsf.conf` file, **mbatchd** does no user group checking, and just accepts the job submission. The parameter **LSF_DATA_SKIP_GROUP_CHECK=Y** affects only the **mbatchd** daemon. User group checking is not affected by how the **CACHE_ACCESS_CONTROL** parameter is configured in the `lsf.datamanager` file.

Examples

The following job requests the data file `/proj/std/model.tar`, which belongs to the user group `design1`:

```
bsub -o %J.out -data "hostA:/proj/std/model.tar" -datagrp "design1" my_job.sh
 Job <1962> is submitted to default queue <normal>.
```

-E

Runs the specified job-based pre-execution command on the execution host before actually running the job.

Categories

properties

Synopsis

`bsub -E "`*pre_exec_command* [*arguments* ...]"

Description

For a parallel job, the job-based pre-execution command runs on the first host selected for the parallel job. If you want the pre-execution command to run on a specific first execution host, specify one or more first execution host candidates at the job level using `-m`, at the queue level with **PRE_EXEC** in `lsb.queues`, or at the application level with **PRE_EXEC** in `lsb.applications`.

If the pre-execution command returns a zero (0) exit code, LSF runs the job on the selected host. Otherwise, the job and its associated pre-execution command goes back to PEND status and is rescheduled. LSF keeps trying to run pre-execution commands and pending jobs. After the pre-execution command runs successfully, LSF runs the job. You must ensure that the pre-execution command can run multiple times without causing side effects, such as reserving the same resource more than once.

The standard input and output for the pre-execution command are directed to the same files as the job. The pre-execution command runs under the same user ID, environment, home, and working directory as the job. If the pre-execution command is not in the user's usual execution path (the $PATH variable), the full path name of the command must be specified.

Note: The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

-Ep

Runs the specified job-based post-execution command on the execution host after the job finishes.

Categories

properties

Synopsis

`bsub -Ep "`*post_exec_command* [*arguments* ...]"

Description

If both application-level (**POST_EXEC** in `lsb.applications`) and job-level post-execution commands are specified, job level post-execution overrides application-level post-execution commands. Queue-level post-execution commands (**POST_EXEC** in `lsb.queues`) run after application-level post-execution and job-level post-execution commands.

Note: The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

-e

Appends the standard error output of the job to the specified file path.

Categories

io

Synopsis

`bsub -e` *error_file*

Description

If the parameter **LSB_STDOUT_DIRECT** in `lsf.conf` is set to Y or y, the standard error output of a job is written to the file you specify as the job runs. If **LSB_STDOUT_DIRECT** is not set, it is written to a temporary file and copied to the specified file after the job finishes. **LSB_STDOUT_DIRECT** is not supported on Windows.

If you use the special character `%J` in the name of the error file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the error file, then `%I` is replaced by the index of the job in the array if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard error output file to `/tmp/`.

If the specified *error_file* path is not accessible, the output will not be stored.

If the specified *error_file* is the same as the *output_file* (as specified by the `-o` option), the *error_file* of the job is NULL and the standard error of the job is stored in the output file.

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

-env

Controls the propagation of the specified job submission environment variables to the execution hosts.

Categories

properties

Synopsis

`bsub -env "none" | "all, [~`*var_name*`[, ~`*var_name*`] ...] [`*var_name*`=`*var_value*`[, `*var_name*`=`*var_value*`] ...]" | "`*var_name*`[=`*var_value*`][, `*var_name*`[=`*var_value*`] ...]"`

Description

Specify a comma-separated list of job submission environment variables to control the propagation to the execution hosts.

- Specify none with no other variables to submit jobs with no submission environment variables. All environment variables are removed while submitting the job.
- Specify the variable name without a value to propagate the environment variable with its default value.
- Specify the variable name with a value to propagate the environment variable with the specified value to overwrite the default value. The specified value may either be a new value or quote the value of an existing environment variable (unless you are submitting job packs). For example:

  In UNIX, `fullpath=/tmp/:$filename` appends `/tmp/` to the beginning of the *filename* environment variable and assigns this new value to the *fullpath* environment variable. Use a colon (`:`) to separate multiple environment variables.

  In Windows, `fullpath=\Temp\:%filename%` appends `\Temp\` to the beginning of the *filename* environment variable and assigns this new value to the *fullpath* environment variable. Use a semicolon (`;`) to separate multiple environment variables.

  The shell under which you submitted the job will parse the quotation marks.
- Specify `all` at the beginning of the list to propagate all existing submission environment variables to the execution hosts. You may also assign values to specific environment variables.

  For example, `-env "all, var1=value1, var2=value2"` submits jobs with all the environment variables, but with the specified values for the *var1* and *var2* environment variables.
- When using the `all` keyword, add ~ to the beginning of the variable name and the environment variable is not propagated to the execution hosts.

The environment variable names cannot contain the following words and symbols: "`none`", "`all`", comma (`,`), tilde (~), equals sign (=), double quotation mark (") and single quotation mark (').

The variable value can contain a tilde (~) and a comma (`,`). However, if the value contains a comma (`,`), the entire value must be enclosed in single quotation marks. For example:

```
bsub -env "TEST='A, B' "
```

An **esub** can change the `-env` environment variables by writing them to the file specified by the **LSB_SUB_MODIFY_FILE** or **LSF_SUB4_SUB_ENV_VARS** environment variables. If both environment variables are specified, **LSF_SUB_MODIFY_FILE** takes effect.

When `-env` is not specified with **bsub**, the default value is `-env "all"` (that is, all environment variables are submitted with the default values).

The entire argument for the `-env` option may contain a maximum of 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

If `-env` conflicts with `-L`, the value of `-L` takes effect.

The following environment variables are not propagated to execution hosts because they are only used in the submission host:

- HOME, LS_JOBPID, LSB_ACCT_MAP, LSB_EXIT_PRE_ABORT, LSB_EXIT_REQUEUE, LSB_EVENT_ATTRIB, LSB_HOSTS, LSB_INTERACTIVE, LSB_INTERACTIVE_SSH, LSB_INTERACTIVE_TTY, LSB_JOBFILENAME, LSB_JOBGROUP, LSB_JOBID, LSB_JOBNAME, LSB_JOB_STARTER, LSB_QUEUE, LSB_RESTART, LSB_TRAPSIGS, LSB_XJOB_SSH, LSF_VERSION, PWD, USER, VIRTUAL_HOSTNAME, and all variables with starting with LSB_SUB_
- Environment variables about non-interactive jobs: TERM, TERMCAP
- Windows-specific environment variables: COMPUTERNAME, COMSPEC, NTRESKIT, OS2LIBPATH, PROCESSOR_ARCHITECTURE, PROCESSOR_IDENTIFIER, PROCESSOR_LEVEL, PROCESSOR_REVISION, SYSTEMDRIVE, SYSTEMROOT, TEMP, TMP

The following environment variables do not take effect on the execution hosts: LSB_DEFAULTPROJECT, LSB_DEFAULT_JOBGROUP, LSB_TSJOB_ENVNAME, LSB_TSJOB_PASSWD, LSF_DISPLAY_ALL_TSC, LSF_JOB_SECURITY_LABEL, LSB_DEFAULT_USERGROUP, LSB_DEFAULT_RESREQ, LSB_DEFAULTQUEUE, BSUB_CHK_RESREQ, LSB_UNIXGROUP, LSB_JOB_CWD

-eo

Overwrites the standard error output of the job to the specified file path.

Categories

io

Synopsis

`bsub -eo` *error_file*

Description

If the parameter **LSB_STDOUT_DIRECT** in `lsf.conf` is set to Y or y, the standard error output of a job is written to the file you specify as the job runs, which occurs every time the job is submitted with the overwrite option, even if it is requeued manually or by the system. If **LSB_STDOUT_DIRECT** is not set, it is written to a temporary file and copied to the specified file after the job finishes. **LSB_STDOUT_DIRECT** is not supported on Windows.

If you use the special character `%J` in the name of the error file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the error file, then `%I` is replaced by the index of the job in the array if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard error output file to `/tmp/`.

If the specified *error_file* path is not accessible, the output will not be stored.

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

-eptl

Specifies the eligible pending time limit for the job.

Categories

limit

Synopsis

`bsub -eptl` [*hour*`:`]*minute*

Description

`TRACK_ELIGIBLE_PENDINFO=Y` must be defined in the `lsb.params` file to use the **bsub -eptl** option.

LSF sends the job-level eligible pending time limit configuration to IBM Spectrum LSF RTM (LSF RTM), which handles the alarm and triggered actions such as user notification (for example, notifying the user that submitted the job and the LSF administrator) and job control actions (for example, killing the job). LSF RTM compares the job's eligible pending time to the eligible pending time limit, and if the job is in an eligible pending state for longer than this specified time limit, LSF RTM triggers the alarm and actions. Without LSF RTM, LSF shows the pending time limit and takes no action on the pending time limit.

In MultiCluster job forwarding mode, the job's eligible pending time limit is ignored in the execution cluster, while the submission cluster merges the job's queue-, application-, and job-level eligible pending time limit according to local settings.

The eligible pending time limit is in the form of [*hour*`:`]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

To specify a pending time limit or eligible pending time limit at the queue or application level, specify the **PEND_TIME_LIMIT** or **ELIGIBLE_PEND_TIME_LIMIT** parameters in the `lsb.queues` or `lsb.applications` files.

The job-level eligible pending time limit specified here overrides any application-level or queue-level limits specified (**ELIGIBLE_PEND_TIME_LIMIT** in `lsb.applications` and `lsb.queues`).

-ext

Specifies application-specific external scheduling options for the job.

Categories

schedule

Synopsis

`bsub -ext[sched] "`*external_scheduler_options*`"`

Description

To enable jobs to accept external scheduler options, set `LSF_ENABLE_EXTSCHEDULER=y` in `lsf.conf`.

You can abbreviate the `-extsched` option to `-ext`.

You can specify only one type of external scheduler option in a single `-extsched` string.

For example, Linux hosts and SGI VCPUSET hosts running CPUSET can exist in the same cluster, but they accept different external scheduler options. Use external scheduler options to define job requirements for either Linux or CPUSET, but not both. Your job runs either on Linux hosts or CPUSET hosts. If external scheduler options are not defined, the job may run on a Linux host but it does not run on a CPUSET host.

The options set by `-extsched` can be combined with the queue-level **MANDATORY_EXTSCHED** or **DEFAULT_EXTSCHED** parameters. If `-extsched` and **MANDATORY_EXTSCHED** set the same option, the **MANDATORY_EXTSCHED** setting is used. If `-extsched` and **DEFAULT_EXTSCHED** set the same options, the `-extsched` setting is used.

Use **DEFAULT_EXTSCHED** in `lsb.queues` to set default external scheduler options for a queue.

To make certain external scheduler options mandatory for all jobs submitted to a queue, specify **MANDATORY_EXTSCHED** in `lsb.queues` with the external scheduler options you need or your jobs.

-F

Sets a per-process (soft) file size limit for each of the processes that belong to the job.

Categories

limit

Synopsis

`bsub -F` *file_limit*

Description

For more information, see **getrlimit**(2). The limit is specified in KB.

If a job process attempts to write to a file that exceeds the file size limit, then that process is sent a SIGXFSZ signal. The SIGXFSZ signal normally terminates the process.

-f

Copies a file between the local (submission) host and the remote (execution) host.

Categories

io

Synopsis

`bsub -f " local_file operator [remote_file]" ...`

Description

Specify absolute or relative paths, including the file names. You should specify the remote file as a file name with no path when running in non-shared systems.

If the remote file is not specified, it defaults to the local file, which must be given. Use multiple `-f` options to specify multiple files.

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

operator

An operator that specifies whether the file is copied to the remote host, or whether it is copied back from the remote host. The operator must be surrounded by white space.

The following describes the operators:

> Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists.

< Copies the remote file to the local file after the job completes. Overwrites the local file if it exists.

<< Appends the remote file to the local file after the job completes. The local file must exist.

>< Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists. Then copies the remote file to the local file after the job completes. Overwrites the local file.

<> Copies the local file to the remote file before the job starts. Overwrites the remote file if it exists. Then copies the remote file to the local file after the job completes. Overwrites the local file.

All of the above involve copying the files to the output directory defined in **DEFAULT_JOB_OUTDIR** or with **bsub -outdir** instead of the submission directory, as long as the path is relative. The output directory is created at the start of the job, and also applies to jobs that are checkpointed, migrated, requeued or rerun.

If you use the `-i` *input_file* option, then you do not have to use the `-f` option to copy the specified input file to the execution host. LSF does this for you, and removes the input file from the execution host after the job completes.

If you use the `-o` *out_file*,`-e` *err_file*, `-oo` *out_file*, or the `-eo` *err_file* option, and you want the specified file to be copied back to the submission host when the job completes, then you must use the `-f` option.

If the submission and execution hosts have different directory structures, you must make sure that the directory where the remote file and local file are placed exists.

If the local and remote hosts have different file name spaces, you must always specify relative path names. If the local and remote hosts do not share the same file system, you must make sure that the directory containing the remote file exists. It is recommended that only the file name be given for the remote file when running in heterogeneous file systems. This places the file in the job's current working directory. If the file is shared between the submission and execution hosts, then no file copy is performed.

LSF uses **lsrcp** to transfer files (see **lsrcp**(1) command). **lsrcp** contacts RES on the remote host to perform the file transfer. If RES is not available, **rcp** is used (see **rcp**(1)). The user must make sure that the **rcp** binary is in the user's $PATH on the execution host.

Jobs that are submitted from LSF client hosts should specify the `-f` option only if **rcp** is allowed. Similarly, **rcp** must be allowed if account mapping is used.

-freq

Specifies a CPU frequency for a job.

Categories

resource

Synopsis

`bsub -freq` *numberUnit*

Description

The submission value will overwrite the application profile value and the application profile value will overwrite the queue value. The value is float and should be specified with SI units (GHz, MHz, KHz). If no units are specified GHz is the default.

-G

For fairshare scheduling. Associates the job with the specified group or excludes the job from the specified groups.

Categories

schedule

Synopsis

`bsub -G` *user_group*

`bsub -G "~`*user_group* ...`"`

Description

Specify any group that you belong to. You must be a direct member of the specified user group.

If you belong to multiple groups, use a tilde (~) to specify any fairshare groups from which you want to exclude for the job. Use a space-separated list of tildes (~) to exclude multiple groups. If you exclude a middle node from a fairshare tree, the descendant groups of that middle node are excluded along the specific path under the middle node.

For example, if there are two paths in the fairshare tree to userA, which are `/groupA/groupB/userA` and `/groupB/userA`, and you specify **bsub -G "~groupA"**, userA can still be associated with the path `/groupB/userA`.

If you are associating the job with a group, you cannot also exclude a group. For example, you cannot specify `bsub -G "groupC ~groupD"` to associate a job with groupC and to exclude the job from groupD.

If **ENFORCE_ONE_UG_LIMITS** is enabled in `lsb.params`, using the `-G` option enforces any limits placed on the specified user group only even if the user or user group belongs to more than one group and does not enforce limits on excluded user groups.

If **ENFORCE_ONE_UG_LIMITS** is disabled in `lsb.params` (default), using the `-G` option enforces the strictest limit that is set on any of the groups that the user or user group belongs to.

Examples

If `ENFORCE_ONE_UG_LIMIT=Y` is enabled, the following commands have the following effects:

- **bsub -G "group1"**

Associates the job with the group1 fairshare group. If ENFORCE_ONE_UG_LIMIT=Y is enabled in the lsb.params file, limits apply only to group1 and its parents.

- **bsub -G "~group1 ~group2"**

  Ensures that the job is never associated with group1 or group2. If ENFORCE_ONE_UG_LIMIT=Y is enabled in the lsb.params file, limits do not apply to group1 or group2.

-g

Submits jobs in the specified job group.

Categories

properties

Synopsis

bsub -g *job_group_name*

Description

The job group does not have to exist before submitting the job. For example:

```
bsub -g /risk_group/portfolio1/current myjob
Job <105> is submitted to default queue.
```

Submits myjob to the job group /risk_group/portfolio1/current.

If group /risk_group/portfolio1/current exists, job 105 is attached to the job group.

Job group names can be up to 512 characters long.

If group /risk_group/portfolio1/current does not exist, LSF checks its parent recursively, and if no groups in the hierarchy exist, all three job groups are created with the specified hierarchy and the job is attached to group.

You can use -g with -sla. All jobs in a job group attached to a service class are scheduled as SLA jobs. It is not possible to have some jobs in a job group not part of the service class. Multiple job groups can be created under the same SLA. You can submit additional jobs to the job group without specifying the service class name again. You cannot use job groups with resource-based SLAs that have guarantee goals.

For example, the following attaches the job to the service class named opera, and the group /risk_group/portfolio1/current:

```
bsub -sla opera -g /risk_group/portfolio1/current myjob
```

To submit another job to the same job group, you can omit the SLA name:

```
bsub -g /risk_group/portfolio1/current myjob2
```

-gpu

Specifies properties of GPU resources required by the job.

Categories

resources

Synopsis

bsub -gpu - | "[num=*num_gpus*[/task | host]] [:mode=shared | exclusive_process] [:mps=yes[,shared] | no | per_socket[,shared] | per_gpu[,shared]] [:j_exclusive=yes | no] [:aff=yes | no] [:gmodel=*model_name*[-*mem_size*]] [:gtile=*tile_num*|'!'] [:gmem=*mem_value*] [:nvlink=yes]"

Description

-

Specifies that the job does not set job-level GPU requirements. Use the hyphen with no letter to set the effective GPU requirements, which are defined at the cluster, queue, or application profile level.

If a GPU requirement is specified at the cluster, queue, and application profile level, each option (`num`, `mode`, `mps`, `j_exclusive`, `gmodel`, `gtile`, `gmem`, and `nvlink`) of the GPU requirement is merged separately. Application profile level overrides queue level, which overrides the cluster level default GPU requirement.

If there are no GPU requirements defined in the cluster, queue, or application level, the default value is `"num=1:mode=shared:mps=no:j_exclusive=no"`.

num=*num_gpus*[/task | host]

The number of physical GPUs required by the job. By default, the number is per host. You can also specify that the number is per task by specifying `/task` after the number.

If you specified that the number is per task, the configuration of the `ngpus_physical` resource in the `lsb.resources` file is set to PER_TASK, or the **RESOURCE_RESERVE_PER_TASK=Y** parameter is set in the `lsb.params` file, this number is the requested count per task.

mode=shared | exclusive_process

The GPU mode when the job is running, either `shared` or `exclusive_process`. The default mode is `shared`.

The `shared` mode corresponds to the NVIDIA DEFAULT compute mode. The `exclusive_process` mode corresponds to the NVIDIA EXCLUSIVE_PROCESS compute mode.

mps=yes[,shared] | per_socket[,shared] | per_gpu[,shared] | no

Enables or disables the NVIDIA Multi-Process Service (MPS) for the GPUs that are allocated to the job. Using MPS effectively causes the EXCLUSIVE_PROCESS mode to behave like the DEFAULT mode for all MPS clients. MPS always allows multiple clients to use the GPU through the MPS server.

MPS is useful for both shared and exclusive process GPUs, and allows more efficient sharing of GPU resources and better GPU utilization. See the NVIDIA documentation for more information and limitations.

When using MPS, use the EXCLUSIVE_PROCESS mode to ensure that only a single MPS server is using the GPU, which provides additional insurance that the MPS server is the single point of arbitration between all CUDA process for that GPU.

You can also enable MPS daemon sharing by adding the **share** keyword with a comma and no space (for example, `mps=yes,shared` enables MPS daemon sharing on the host). If sharing is enabled, all jobs that are submitted by the same user with the same resource requirements share the same MPS daemon on the host, socket, or GPU.

LSF starts MPS daemons on a per-host, per-socket, or per-GPU basis depending on value of the **mps** keyword:

- If `mps=yes` is set, LSF starts one MPS daemon per host per job.

  When share is enabled (that is, if `mps=yes,shared` is set), LSF starts one MPS daemon per host for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon on the host.
- If `mps=per_socket` is set, LSF starts one MPS daemon per socket per job. When enabled with share (that is, if `mps=per_socket,shared` is set), LSF starts one MPS daemon per socket for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon for the socket.
- If `mps=per_gpu` is set, LSF starts one MPS daemon per GPU per job. When enabled with share (that is, if `mps=per_gpu,shared` is set), LSF starts one MPS daemon per GPU for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon for the GPU.

Note: Using EXCLUSIVE_THREAD mode with MPS is not supported and might cause unexpected behavior.

j_exclusive=yes | no
Specifies whether the allocated GPUs can be used by other jobs. When the mode is set to `exclusive_process`, the `j_exclusive=yes` option is set automatically.

aff=yes | no
Specifies whether to enforce strict GPU-CPU affinity binding. If set to `no`, LSF relaxes GPU affinity while maintaining CPU affinity. By default, `aff=yes` is set to maintain strict GPU-CPU affinity binding.

gmodel=*model_name*[-*mem_size*]
Specifies the GPUs with the specific model name and, optionally, its total GPU memory. By default, LSF allocates the GPUs with the same model, if available.

The **gmodel** keyword supports the following formats:

gmodel=*model_name*
Requests GPUs with the specified brand and model name (for example, TeslaK80).

gmodel=*short_model_name*
Requests GPUs with a specific brand name (for example, Tesla, Quadro, NVS) or model type name (for example, K80, P100).

gmodel=*model_name-mem_size*
Requests GPUs with the specified brand name and total GPU memory size. The GPU memory size consists of the number and its unit, which includes `M`, `G`, `T`, `MB`, `GB`, and `TB` (for example, `12G`).

To find the available GPU model names on each host, run the **lsload -gpuload**, **lshosts -gpu**, or **bhosts -gpu** commands. The model name string does not contain space characters. In addition, the slash (/) and hyphen (-) characters are replaced with the underscore character (_). For example, the GPU model name "`Tesla C2050 / C2070`" is converted to "`TeslaC2050_C2070`" in LSF.

gmem=*mem_value*

Specify the GPU memory on each GPU required by the job. The format of *mem_value* is the same to other resource value (for example, `mem` or `swap`) in the rusage section of the job resource requirements (`-R`).

gtile=! | *tile_num*
Specifies the number of GPUs per socket. Specify an number to explicitly define the number of GPUs per socket on the host, or specify an exclamation mark (`!`) to enable LSF to automatically calculate the number, which evenly divides the GPUs along all sockets on the host. LSF guarantees the **gtile** requirements even for affinity jobs. This means that LSF might not allocate the GPU's affinity to the allocated CPUs when the **gtile** requirements cannot be satisfied.

If the **gtile** keyword is not specified for an affinity job, LSF attempts to allocate enough GPUs on the sockets that allocated GPUs. If there are not enough GPUs on the optimal sockets, jobs cannot go to this host.

If the **gtile** keyword is not specified for a non-affinity job, LSF attempts to allocate enough GPUs on the same socket. If this is not available, LSF might allocate GPUs on separate GPUs.

nvlink=yes
Enables the job enforcement for NVLink connections among GPUs. LSF allocates GPUs with NVLink connections in force.

By default, LSF can allocate GPUs without NVLink connections when there are not enough GPUs with NVLink connections.

The syntax of the GPU requirement in the `-gpu` option is the same as the syntax in the **LSB_GPU_REQ** parameter in the `lsf.conf` file and the **GPU_REQ** parameter in the `lsb.queues` and `lsb.applications` files.

Note: The **bjobs** output does not show `aff=yes` even if you specify `aff=yes` in the **bsub -gpu** option.

The **LSB_GPU_NEW_SYNTAX=extend** or **LSB_GPU_NEW_SYNTAX=Y** parameter must be defined in the `lsf.conf` file to enable the `-gpu` option and **GPU_REQ** parameter syntax. When the **LSB_GPU_NEW_SYNTAX=extend** parameter is set, you cannot use the GPU resources `ngpus_shared`, `ngpus_excl_t` and `ngpus_excl_p` as job resource requirements.

If the **GPU_REQ_MERGE** parameter is defined as Y or y in the `lsb.params` file and a GPU requirement is specified at multiple levels (at least two of the default cluster, queue, application profile, or job level requirements), each option of the GPU requirement is merged separately. Job level overrides application level, which overrides queue level, which overrides the default cluster GPU requirement. For example, if the `mode` option of the GPU requirement is defined on the `-gpu` option, and the `mps` option is defined in the queue, the mode of job level and the `mps` value of queue is used.

If the **GPU_REQ_MERGE** parameter is not defined as Y or y in the `lsb.params` file and a GPU requirement is specified at multiple levels (at least two of the default cluster, queue, application profile, or job level requirements), the entire GPU requirement string is replaced. The entire job level GPU requirement string overrides application level, which overrides queue level, which overrides the default GPU requirement.

The **esub** parameter **LSB_SUB4_GPU_REQ** modifies the value of the `-gpu` option.

LSF selects the GPU that meets the topology requirement first. If the GPU mode of the selected GPU is not the requested mode, LSF changes the GPU to the requested mode. For example, if LSF allocates an `exclusive_process` GPU to a job that needs a shared GPU, LSF changes the GPU mode to shared before the job starts and then changes the mode back to `exclusive_process` when the job finishes.

The GPU requirements are converted to `rusage` resource requirements for the job. For example, `num=2` is converted to `rusage[ngpus_physical=2]`. Use the **bjobs**, **bhist**, and **bacct** commands to see the merged resource requirement.

There might be complex GPU requirements that the **bsub -gpu** option and **GPU_REQ** parameter syntax cannot cover, including compound GPU requirements (for different GPU requirements for jobs on different hosts, or for different parts of a parallel job) and alternate GPU requirements (if more than one set of GPU requirements might be acceptable for a job to run). For complex GPU requirements, use the **bsub -R** command option, or the **RES_REQ** parameter in the `lsb.applications` or `lsb.queues` file to define the resource requirement string.

Note: You can define the `mode`, `j_exclusive`, and `mps` options only with the `-gpu` option, the **LSB_GPU_REQ** parameter in the `lsf.conf` file, or the **GPU_REQ** parameter in the `lsb.queues` or `lsb.applications` files. You cannot use these options with the `rusage` resource requirement string in the **bsub -R** command option or the **RES_REQ** parameter in the `lsb.queues` or `lsb.applications` files.

Examples

The following job request does not override the effective GPU requirement with job-level GPU requirements. The effective GPU requirement is merged together from GPU requirements that are specified at the cluster, queue, and application profile level. Application profile level overrides queue level, which overrides the cluster level default GPU requirement.

```
bsub -gpu - ./app
```

The following job requires 2 EXCLUSIVE_PROCESS mode GPUs and starts MPS before running the job.

```
bsub -gpu "num=2:mode=exclusive_process:mps=yes" ./app
```

The following job requires 2 DEFAULT mode GPUs and uses them exclusively. The two GPUs cannot be used by other jobs even though the mode is shared.

```
bsub -gpu "num=2:mode=shared:j_exclusive=yes" ./app
```

The following job uses 3 DEFAULT mode GPUs and shares them with other jobs.

```
bsub -gpu "num=3:mode=shared:j_exclusive=no" ./app
```

The following job uses 4 EXCLUSIVE_PROCESS GPUs that cannot be used by other jobs. The `j_exclusive` option defaults to `yes` for this job.

```
bsub -gpu "num=4:mode=exclusive_process" ./app
```

The following job requires two tasks. Each task requires 2 EXCLUSIVE_PROCESS GPUs on two hosts. The GPUs are allocated in the same NUMA as the allocated CPU.

```
bsub -gpu "num=2:mode=exclusive_process" -n2 -R "span[ptile=1] affinity[core(1)]" ./app
```

See also

LSB_GPU_REQ and **LSB_GPU_NEW_SYNTAX** parameters in the `lsf.conf` file and the **GPU_REQ** parameter in the `lsb.queues` and `lsb.applications` files

-H

Holds the job in the PSUSP state when the job is submitted.

Categories

schedule

Synopsis

`bsub -H`

Description

The job is not scheduled until you tell the system to resume the job (see **bresume(1)**).

-hl

Enables job-level host-based memory and swap limit enforcement on systems that support **Linux** cgroups.

Categories

limit

Synopsis

`bsub -hl`

Description

When `-hl` is specified, a memory limit specified at the job level by `-M` or by **MEMLIMIT** in `lsb.queues` or `lsb.applications` is enforced by the Linux cgroup subsystem on a per-job basis on each host. Similarly, a swap limit specified at the job level by `-v` or by **SWAPLIMIT** in `lsb.queues` or `lsb.applications` is enforced by the Linux cgroup subsystem on a per-job basis on each host. Host-based memory and swap limits are enforced regardless of the number of tasks running on the execution host. The `-hl` option only applies to memory and swap limits; it does not apply to any other resource usage limits.

LSB_RESOURCE_ENFORCE="memory" must be specified in `lsf.conf` for host-based memory and swap limit enforcement with the `-hl` option to take effect. If no memory or swap limit is specified for the job (the merged limit for the job, queue, and application profile, if specified), or **LSB_RESOURCE_ENFORCE="memory"** is not specified, a host-based memory limit is not set for the job.

When **LSB_RESOURCE_ENFORCE="memory"** is configured in `lsf.conf`, and memory and swap limits are specified for the job, but `-hl` is *not* specified, memory and swap limits are calculated and enforced as a multiple of the number of tasks running on the execution host.

-hostfile

Submits a job with a user-specified host file.

Categories

resource

Synopsis

`bsub -hostfile` *file_path*

Conflicting options

Do not use with the following options: `-ext`, `-n`, `-m`, `-R` *res-req*.

Description

When submitting a job, you can point the job to a file that specifies hosts and number of slots for job processing.

For example, some applications (typically when benchmarking) run best with a very specific geometry. For repeatability (again, typically when benchmarking) you may want it to always run it on the same hosts, using the same number of slots.

The user-specified host file specifies a host and number of slots to use per task, resulting in a rank file.

The `-hostfile` option allows a user to submit a job, specifying the path of the user-specified host file:

```
bsub -hostfile "spec_host_file"
```

Important:

- Do not use a user-specified host file if you have enabled task geometry as it may cause conflicts and jobs may fail.
- Alternatively, if resources are not available at the time that a task is ready a job may not run smoothly. Consider using advance reservation instead of a user-specified host file, to ensure reserved slots are available.

Any user can create a user-specified host file. It must be accessible by the user from the submission host. It lists one host per line. The format is as follows:

```
# This is a user-specified host file
<host_name1>   [<# slots>]
<host_name2>   [<# slots>]
<host_name1>   [<# slots>]
<host_name2>   [<# slots>]
<host_name3>   [<# slots>]
<host_name4>   [<# slots>]
```

The following rules apply to the user-specified host file:

- Insert comments starting with the # character.
- Specifying the number of slots for a host is optional. If no slot number is indicated, the default is 1.
- A host name can be either a host in a local cluster or a host leased-in from a remote cluster (*host_name*@*cluster_name*).
- A user-specified host file should contain hosts from the same cluster only.
- A host name can be entered with or without the domain name.
- Host names may be used multiple times and the order entered represents the placement of tasks. For example:

```
#first three tasks
host01                      3
#fourth tasks
```

```
host02
#next three tasks
host03                      3
```

The resulting rank file is made available to other applications (such as MPI).

The **LSB_DJOB_RANKFILE** environment variable is generated from the user-specified host file. If a job is not submitted with a user-specified host file then **LSB_DJOB_RANKFILE** points to the same file as **LSB_DJOB_HOSTFILE**.

The **esub** parameter **LSB_SUB4_HOST_FILE** reads and modifies the value of the `-hostfile` option.

The following is an example of a user-specified host file that includes duplicate host names:

```
user1: cat ./user1_host_file
# This is my user-specified host file for job242
host01   3
host02
host03   3
host01
host02   2
```

This user-specified host file tells LSF to allocate 10 slots in total (4 slots on `host01`, 3 slots on `host02`, and 3 slots on `host03`). Each line represents the order of task placement.

Duplicate host names are combined, along with the total number of slots for a host name and the results are used for scheduling (whereas **LSB_DJOB_HOSTFILE** groups the hosts together) and for **LSB_MCPU_HOSTS**. **LSB_MCPU_HOSTS** represents the job allocation.

The result is the following:

LSB_DJOB_RANKFILE:

```
host01
host01
host01
host02
host03
host03
host03
host01
host02
host02
```

LSB_DJOB_HOSTFILE:

```
host01
host01
host01
host01
host02
host02
host02
host03
host03
host03
```

```
LSB_MCPU_HOSTS = host01 4 host02 3 host03 3
```

-I

Submits an interactive job.

Categories

properties

Synopsis

```
bsub -I [-tty]
```

Conflicting options

Do not use with the following options: `-Ip`, `-IS`, `-ISp`, `-ISs`, `-Is`, `-IX`, `-K`.

Description

Submits an interactive job. A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the `-N` option.

If the `-i` *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the `-e` *err_file* option is specified, sends the job's standard error to the specified error file.

If used with **-tty**, also displays output/error (except pre-exec output/error) on the screen.

Interactive jobs cannot be checkpointed.

Interactive jobs are not rerunnable (**bsub -r**).

Examples

```
bsub -I ls
```

Submit an interactive job that displays the output of **ls** at the user's terminal.

-Ip

Submits an interactive job and creates a pseudo-terminal when the job starts.

Categories

properties

Synopsis

```
bsub -Ip [-tty]
```

Conflicting options

Do not use with the following options: `-I`, `-IS`, `-ISp`, `-ISs`, `-Is`, `-IX`, `-K`.

Description

Some applications (for example, **vi**) require a pseudo-terminal in order to run correctly.

Options that create a pseudo-terminal are not supported on Windows. Since the `-Ip` option creates a pseudo-terminal, it is not supported on Windows.

A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the `-N` option.

If the `-i` *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the `-e` *err_file* option is specified, sends the job's standard error to the specified error file.

If used with **-tty**, also displays output/error (except pre-exec output/error) on the screen.

Interactive jobs cannot be checkpointed.

Interactive jobs are not rerunnable (**bsub -r**).

Examples

```
bsub -Ip vi myfile
```

Submit an interactive job to edit `myfile`.

-IS

Submits an interactive job under a secure shell (**ssh**).

Categories

properties

Synopsis

```
bsub -IS [-tty]
```

Conflicting options

Do not use with the following options: `-I`, `-Ip`, `-ISp`, `-ISs`, `-Is`, `-IX`, `-K`.

Description

A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the `-N` option.

If the `-i` *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the `-e` *err_file* option is specified, sends the job's standard error to the specified error file.

If used with `-tty`, also displays output/error on the screen.

Interactive jobs cannot be checkpointed.

Interactive jobs are not rerunnable (**bsub -r**).

-ISp

Submits an interactive job under a secure shell (**ssh**) and creates a pseudo-terminal when the job starts.

Categories

properties

Synopsis

```
bsub -ISp [-tty]
```

Conflicting options

Do not use with the following options: `-I`, `-Ip`, `-IS`, `-ISs`, `-Is`, `-IX`, `-K`.

Description

Some applications (for example, **vi**) require a pseudo-terminal in order to run correctly. The options that create a pseudo-terminal are not supported on Windows.

A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the `-N` option.

If the `-i` *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the `-e` *err_file* option is specified, sends the job's standard error to the specified error file.

If used with `-tty`, also displays output/error on the screen.

Interactive jobs cannot be checkpointed.

Interactive jobs are not rerunnable (**bsub -r**).

-ISs

Submits an interactive job under a secure shell (**ssh**) and creates a pseudo-terminal with shell mode support when the job starts.

Categories

properties

Synopsis

```
bsub -ISs [-tty]
```

Conflicting options

Do not use with the following options: `-I`, `-Ip`, `-IS`, `-ISp`, `-Is`, `-IX`, `-K`.

Description

Some applications (for example, **vi**) require a pseudo-terminal in order to run correctly. The options that create a pseudo-terminal are not supported on Windows.

Specify this option to add shell mode support to the pseudo-terminal for submitting interactive shells, or applications that redefine the CTRL-C and CTRL-Z keys (for example, `jove`).

A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the `-N` option.

If the `-i` *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the `-e` *err_file* option is specified, sends the job's standard error to the specified error file.

If used with `-tty`, also displays output/error on the screen.

Interactive jobs cannot be checkpointed.

Interactive jobs are not rerunnable (**bsub -r**).

-Is

Submits an interactive job and creates a pseudo-terminal with shell mode when the job starts.

Categories

properties

Synopsis

```
bsub -Is [-tty]
```

Conflicting options

Do not use with the following options: `-I`, `-Ip`, `-IS`, `-ISp`, `-ISs`, `-IX`, `-K`.

Description

Some applications (for example, **vi**) require a pseudo-terminal in order to run correctly.

Options that create a pseudo-terminal are not supported on Windows. Since the `-Is` option creates a pseudo-terminal, it is not supported on Windows.

Specify this option to add shell mode support to the pseudo-terminal for submitting interactive shells, or applications that redefine the CTRL-C and CTRL-Z keys (for example, `jove`).

A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the `-N` option.

If the `-i` *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the `-e` *err_file* option is specified, sends the job's standard error to the specified error file.

If used with **-tty**, also displays output/error (except pre-exec output/error) on the screen.

Interactive jobs cannot be checkpointed.

Interactive jobs are not rerunnable (**bsub -r**).

Examples

```
bsub -Is csh
```

Submit an interactive job that starts **csh** as an interactive shell.

-IX

Submits an interactive X-Window job.

Categories

properties

Synopsis

```
bsub -IX [-tty]
```

Conflicting options

Do not use with the following options: `-I`, `-Ip`, `-IS`, `-ISp`, `-ISs`, `-Is`, `-K`.

Description

A new job cannot be submitted until the interactive job is completed or terminated.

Sends the job's standard output (or standard error) to the terminal. Does not send mail to you when the job is done unless you specify the `-N` option.

The session between X-client and X-server is encrypted; the session between the execution host and submission host is also encrypted. The following must be satisfied:

- openssh must be installed and sshd must be running on the X-server
- `xhost + localhost` or `xhost + displayhost.domain.com` on the X-server
- ssh must be configured to run without a password or passphrase (`$HOME/.ssh/authorized_keys` must be set up)

  Note: In most cases ssh can be configured to run without a password by copying `id_rsa.pub` as `authorized_keys` with permission 600 (-rw-r--r--). Test by manually running **ssh host.domain.com** between the two hosts both ways and confirm there are no prompts using fully qualified host names.

If the `-i` *input_file* option is specified, you cannot interact with the job's standard input via the terminal.

If the `-o` *out_file* option is specified, sends the job's standard output to the specified output file. If the `-e` *err_file* option is specified, sends the job's standard error to the specified error file.

If used with `-tty`, also displays output/error on the screen.

Interactive jobs cannot be checkpointed.

Interactive jobs are not rerunnable (**bsub -r**).

Troubleshooting

Use the SSH command on the job execution host to connect it securely with the job submission host. If the host fails to connect, you can perform the following steps to troubleshoot.

- Check the SSH version on both hosts. If the hosts have different SSH versions, a message is displayed identifying a protocol version mismatch.
- Check that public and private key pairs are correctly configured.
- Check the domain name.

```
$ ssh –f –L 6000:localhost:6000 domain_name.example.com date
$ ssh –f –L 6000:localhost:6000 domain_name date
```

If these commands return errors, troubleshoot the domain name with the error information returned.

The execution host should connect without passwords and pass phrases; for example:

```
$ ssh sahpia03
$ ssh sahpia03.example.com
```

-i

Gets the standard input for the job from specified file path.

Categories

io

Synopsis

`bsub -i` *input_file*

Conflicting options

Do not use with the `-is` option.

Description

Specify an absolute or relative path. The input file can be any type of file, though it is typically a shell script text file.

You can use the special characters `%J` and `%I` in the name of the input file. `%J` is replaced by the job ID. `%I` is replaced by the index of the job in the array, if the job is a member of an array, otherwise by 0 (zero).

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

If the file exists on the execution host, LSF uses it. Otherwise, LSF attempts to copy the file from the submission host to the execution host. For the file copy to be successful, you must allow remote copy (**rcp**) access, or you must submit the job from a server host where RES is running. The file is copied from the submission host to a temporary file in the directory specified by the **JOB_SPOOL_DIR** parameter in `lsb.params`, or your `$HOME/.lsbatch` directory on the execution host. LSF removes this file when the job completes.

By default, the input file is spooled to LSB_SHAREDIR/*cluster_name*/lsf_indir. If the lsf_indir directory does not exist, LSF creates it before spooling the file. LSF removes the spooled file when the job completes. Use the -is option if you need to modify or remove the input file before the job completes. Removing or modifying the original input file does not affect the submitted job.

JOB_SPOOL_DIR can be any valid path up to a maximum length up to 4094 characters on UNIX and Linux or up to 255 characters for Windows.

JOB_SPOOL_DIR must be readable and writable by the job submission user, and it must be shared by the master host and the submission host. If the specified directory is not accessible or does not exist, **bsub** cannot write to the default directory LSB_SHAREDIR/*cluster_name*/lsf_indir and the job fails.

-is

Gets the standard input for the job from the specified file path, but allows you to modify or remove the input file before the job completes.

Categories

io

Synopsis

bsub -is *input_file*

Conflicting options

Do not use with the -i option.

Description

Specify an absolute or relative path. The input file can be any type of file, though it is typically a shell script text file.

The special characters %J and %I are not valid with the -is option.

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

If the file exists on the execution host, LSF uses it. Otherwise, LSF attempts to copy the file from the submission host to the execution host. For the file copy to be successful, you must allow remote copy (**rcp**) access, or you must submit the job from a server host where RES is running. The file is copied from the submission host to a temporary file in the directory specified by the **JOB_SPOOL_DIR** parameter in lsb.params, or your $HOME/.lsbatch directory on the execution host. LSF removes this file when the job completes.

By default, the input file is spooled to LSB_SHAREDIR/*cluster_name*/lsf_indir. If the lsf_indir directory does not exist, LSF creates it before spooling the file. LSF removes the spooled file when the job completes. The -is option allows you to modify or remove the input file before the job completes. Removing or modifying the original input file does not affect the submitted job.

If **JOB_SPOOL_DIR** is specified, the -is option spools the input file to the specified directory and uses the spooled file as the input file for the job.

JOB_SPOOL_DIR can be any valid path up to a maximum length up to 4094 characters on UNIX and Linux or up to 255 characters for Windows.

JOB_SPOOL_DIR must be readable and writable by the job submission user, and it must be shared by the master host and the submission host. If the specified directory is not accessible or does not exist, **bsub** cannot write to the default directory LSB_SHAREDIR/*cluster_name*/lsf_indir and the job fails.

-J

Assigns the specified name to the job, and, for job arrays, specifies the indices of the job array and optionally the maximum number of jobs that can run at any given time.

Categories

properties

Synopsis

`bsub -J` *job_name* | `-J "`*job_name*`[`*index_list*`]%`*job_slot_limit*`"`

Description

The job name does not need to be unique and can contain up to 4094 characters.

To specify a job array, enclose the index list in square brackets, as shown, and enclose the entire job array specification in quotation marks, as shown. The index list is a comma-separated list whose elements have the syntax [start-end[:step]] where start, end and step are positive integers. If the step is omitted, a step of one is assumed. By default, the job array index starts at one.

By default, the maximum number of jobs in a job array is 1000, which means the maximum size of a job array (that is, the maximum job array index) can never exceed 1000 jobs.

To change the maximum job array value, set **MAX_JOB_ARRAY_SIZE** in `lsb.params` to any positive integer between 1 and 2147483646. The maximum number of jobs in a job array cannot exceed the value set by **MAX_JOB_ARRAY_SIZE**.

You may also use a positive integer to specify the system-wide job slot limit (the maximum number of jobs that can run at any given time) for this job array.

All jobs in the array share the same job ID and parameters. Each element of the array is distinguished by its array index.

After a job is submitted, you use the job name to identify the job. Specify "*job_ID*[*index*]" to work with elements of a particular array. Specify "*job_name*[*index*]" to work with elements of all arrays with the same name. Since job names are not unique, multiple job arrays may have the same name with a different or same set of indices.

Examples

```
bsub -b 20:00 -J my_job_name my_program
```

Submit `my_program` to run after 8 p.m. and assign it the job name `my_job_name`.

-Jd

Assigns the specified description to the job; for job arrays, specifies the same job description for all elements in the job array.

Categories

properties

Synopsis

`bsub -Jd "`*job_description*`"`

Description

The job description does not need to be unique and can contain up to 4094 characters.

After a job is submitted, you can use **bmod** `-Jd` to change the job description for any specific job array element, if required.

-jsdl

Submits a job using a JSDL file that uses the LSF extension to specify job submission options.

Categories

properties

Synopsis

`bsub -jsdl` *file_name*

Conflicting options

Do not use with the following options: `-json`, `-jsdl_strict`, `-yaml`.

Description

LSF provides an extension to the JSDL specification so that you can submit jobs using LSF features not defined in the JSDL standard schema. The JSDL schema (`jsdl.xsd`), the POSIX extension (`jsdl-posix.xsd`), and the LSF extension (`jsdl-lsf.xsd`) are located in the `LSF_LIBDIR` directory.

- To submit a job that uses the LSF extension, use the `-jsdl` option.
- To submit a job that uses only standard JSDL elements and POSIX extensions, use the `-jsdl_strict` option. You can use the `-jsdl_strict` option to verify that your file contains only valid JSDL elements and POSIX extensions. Error messages indicate invalid elements, including:
  - Elements that are not part of the JSDL specification
  - Valid JSDL elements that are not supported in this version of LSF
  - Extension elements that are not part of the JSDL standard and POSIX extension schemas

For more information about submitting jobs using JSDL, including a detailed mapping of JSDL elements to LSF submission options, and a complete list of supported and unsupported elements, refer to *Submitting jobs using JSDL* in *Administering IBM Spectrum LSF*.

If you specify duplicate or conflicting job submission parameters, LSF resolves the conflict by applying the following rules:

1. The parameters that are specified in the command line override all other parameters.
2. The parameters that are specified in the JSDL file override the job script.

-jsdl_strict

Submits a job using a JSDL file that only uses the standard JSDL elements and POSIX extensions to specify job submission options.

Categories

properties

Synopsis

`bsub -jsdl_strict` *file_name*

Conflicting options

Do not use with the following options: `-json`, `-jsdl`, `-yaml`.

Description

LSF provides an extension to the JSDL specification so that you can submit jobs using LSF features not defined in the JSDL standard schema. The JSDL schema (`jsdl.xsd`), the POSIX extension (`jsdl-posix.xsd`), and the LSF extension (`jsdl-lsf.xsd`) are located in the `LSF_LIBDIR` directory.

- To submit a job that uses only standard JSDL elements and POSIX extensions, use the `-jsdl_strict` option. You can use the `-jsdl_strict` option to verify that your file contains only valid JSDL elements and POSIX extensions. Error messages indicate invalid elements, including:
  - Elements that are not part of the JSDL specification
  - Valid JSDL elements that are not supported in this version of LSF
  - Extension elements that are not part of the JSDL standard and POSIX extension schemas
- To submit a job that uses the LSF extension, use the `-jsdl` option.

For more information about submitting jobs using JSDL, including a detailed mapping of JSDL elements to LSF submission options, and a complete list of supported and unsupported elements, refer to *Submitting jobs using JSDL* in *Administering IBM Spectrum LSF*.

If you specify duplicate or conflicting job submission parameters, LSF resolves the conflict by applying the following rules:

1. The parameters that are specified in the command line override all other parameters.
2. The parameters that are specified in the JSDL file override the job script.

-jsm

Enables or disables the IBM Job Step Manager (JSM) daemon for the job.

Categories

properties

Synopsis

```
bsub -jsm y | n | d
```

Description

This option is for LSF jobs that are submitted with the IBM Cluster Systems Manager (CSM) integration package, and defines if the job is a JSM job using the **jsrun** mechanism.

If set to d, enables the JSM daemon in debug mode.

Default

y. JSM daemons are enabled by default.

-K

Submits a job and waits for the job to complete. Sends job status messages to the terminal.

Categories

notify, properties

Synopsis

```
bsub -K
```

Conflicting options

Do not use with the following options: `-I`, `-Ip`, `-IS`, `-ISp`, `-ISs`, `-Is`, `-IX`.

Description

Sends the message "`Waiting for dispatch`" to the terminal when you submit the job. Sends the message "`Job is finished`" to the terminal when the job is done.

If **LSB_SUBK_SHOW_EXEC_HOST** is enabled in `lsf.conf`, also sends the message "`Starting on` *`execution_host`*" when the job starts running on the execution host.

You are not able to submit another job until the job is completed. This is useful when completion of the job is required to proceed, such as a job script. If the job needs to be rerun due to transient failures, **bsub** returns after the job finishes successfully. **bsub** exits with the same exit code as the job so that job scripts can take appropriate actions based on the exit codes. **bsub** exits with value 126 if the job was terminated while pending.

-k

Makes a job checkpointable and specifies the checkpoint directory.

Categories

properties

Synopsis

`bsub -k "`*checkpoint_dir* [`init=`*initial_checkpoint_period*] [*checkpoint_period*] [`method=`*method_name*]"

Description

Specify a relative or absolute path name. The quotes (") are required if you specify a checkpoint period, initial checkpoint period, or custom checkpoint and restart method name.

The job ID and job file name are concatenated to the checkpoint dir when creating a checkpoint file.

Note: The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

When a job is checkpointed, the checkpoint information is stored in *checkpoint_dir*/*job_ID*/*file_name*. Multiple jobs can checkpoint into the same directory. The system can create multiple files.

The checkpoint directory is used for restarting the job (see **brestart**(1)). The checkpoint directory can be any valid path.

Optionally, specifies a checkpoint period in minutes. Specify a positive integer. The running job is checkpointed automatically every checkpoint period. The checkpoint period can be changed using **bchkpnt**. Because checkpointing is a heavyweight operation, you should choose a checkpoint period greater than half an hour.

Optionally, specifies an initial checkpoint period in minutes. Specify a positive integer. The first checkpoint does not happen until the initial period has elapsed. After the first checkpoint, the job checkpoint frequency is controlled by the normal job checkpoint interval.

The **echkpnt.**`method_name` and **erestart.***method_name* programs must be in LSF_SERVERDIR or in the directory specified by **LSB_ECHKPNT_METHOD_DIR** (environment variable or set in `lsf.conf`).

If a custom checkpoint and restart method is already specified with **LSB_ECHKPNT_METHOD** (environment variable or in `lsf.conf`), the method you specify with **bsub -k** overrides this.

Process checkpointing is not available on all host types, and may require linking programs with a special libraries (see `libckpt.a`(3)). LSF invokes **echkpnt** (see **echkpnt**(8)) found in **LSF_SERVERDIR** to checkpoint the job. You can override the default **echkpnt** for the job by defining as environment variables or in `lsf.conf` **LSB_ECHKPNT_METHOD** and **LSB_ECHKPNT_METHOD_DIR** to point to your own **echkpnt**. This allows you to use other checkpointing facilities, including application-level checkpointing.

The checkpoint method directory should be accessible by all users who need to run the custom **echkpnt** and **erestart** programs.

Only running members of a chunk job can be checkpointed.

-L

Initializes the execution environment using the specified login shell.

Categories

properties

Synopsis

`bsub -L` *login_shell*

Description

The specified login shell must be an absolute path. This is not necessarily the shell under which the job is executed.

Login shell is not supported on Windows.

On UNIX and Linux, the file path of the login shell can contain up to 58 characters.

If `-L` conflicts with `-env`, the value of `-L` takes effect.

-Lp

Assigns the job to the specified LSF License Scheduler project.

Categories

schedule

Synopsis

`bsub -Lp` *ls_project_name*

-ln_mem

Specifies the memory limit on the launch node (LN) host for the CSM job.

Categories

resource

Synopsis

`bsub -ln_mem` *mem_size*

Conflicting options

Do not use with the `-csm y` option.

Description

This option is for easy mode LSF job submission to run with the IBM Cluster Systems Manager (CSM) integration package.

-ln_slots

Specifies the number of slots to use on the launch node (LN) host for the CSM job.

Categories

resource

Synopsis
`bsub -ln_slots` *num_slots*

Conflicting options

Do not use with the `-csm y` option.

Description

This option is for easy mode LSF job submission to run with the IBM Cluster Systems Manager (CSM) integration package.

-M

Sets a memory limit for all the processes that belong to the job.

Categories

limit

Synopsis
`-M` *mem_limit* `[!]`

Description

By default, LSF sets a per-process (soft) memory limit for all the processes that belong to the job. To set a hard memory limit, specify an exclamation point (`!`) after the memory limit. If the job reaches the hard memory limit, LSF kills the job as soon as it exceeds the memory limit without waiting for the host memory and swap threshold to be reached.

By default, the limit is specified in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for the limit.

You can use the following units for limits:

- KB or K (kilobytes)
- MB or M (megabytes)
- GB or G (gigabytes)
- TB or T (terabytes)
- PB or P (petabytes)
- EB or E (exabytes)
- ZB or Z (zettabytes)

If the **LSB_MEMLIMIT_ENFORCE** or **LSB_JOB_MEMLIMIT** parameters are set to y in the `lsf.conf` file, or if you set the hard memory limits with an exclamation point (`!`), LSF kills the job when it exceeds the memory limit or if the host memory and swap threshold is reached. Otherwise, LSF passes the memory limit to the operating system. UNIX operating systems that support RUSAGE_RSS for the **setrlimit()** function can apply the memory limit to each process.

The Windows operating system does not support the memory limit at the OS level.

-m

Submits a job to be run on specific hosts, host groups, or compute units.

Categories

resource

Synopsis

bsub -m "*host_name*[[!] | +[*pref_level*]] | *host_group*[[!] | +[*pref_level* | *compute_unit*[[!] | +[*pref_level*]] ..."

bsub -m "*host_name*@*cluster_name*[| +[*pref_level*]] | *host_group*@*cluster_name*[| +[*pref_level* | *compute_unit*@*cluster_name*[| +[*pref_level*]] ..."

Description

By default, if multiple hosts are candidates, runs the job on the least-loaded host.

When a compute unit requirement is specified along with a host, host group, or compute unit preference, these preferences will affect the order of a compute unit. The highest preference of hosts within the compute unit is taken as the preference of this compute unit. Thus, the compute unit containing the highest preference host will be considered first. In addition, the job will be rejected unless:

- A host in the list belongs to a compute unit, and
- A host in the first execution list belongs to a compute unit.

When used with a compound resource requirement, the first host allocated must satisfy the simple resource requirement string appearing first in the compound resource requirement.

To change the order of preference, put a plus (+) after the names of hosts or host groups that you would prefer to use, optionally followed by a preference level. For preference level, specify a positive integer, with higher numbers indicating greater preferences for those hosts. For example, `-m "hostA groupB+2 hostC+1"` indicates that `groupB` is the most preferred and `hostA` is the least preferred.

The keyword `others` can be specified with or without a preference level to refer to other hosts not otherwise listed. The keyword `others` must be specified with at least one host name or host group, it cannot be specified by itself. For example, `-m "hostA+ others"` means that `hostA` is preferred over all other hosts.

If you also use `-q`, the specified queue must be configured to include at least a partial list of the hosts in your host list. Otherwise, the job is not submitted. To find out what hosts are configured for the queue, use **bqueues -l**.

If the host group contains the keyword `all`, LSF dispatches the job to any available host, even if the host is not defined for the specified queue.

To display configured host groups and compute units, use **bmgroup**.

For the job forwarding model with the LSF multicluster capability, you cannot specify a remote host by name.

For the job forwarding model with the LSF multicluster capability, specify a remote host using *host_name*@*cluster_name* and change the order of preference by using the plus (+) sign.

For parallel jobs, specify first execution host candidates when you want to ensure that a host has the required resources or runtime environment to handle processes that run on the first execution host.

To specify one or more hosts or host groups as first execution host candidates, add an exclamation point (`!`) after the host name, as shown in the following example:

```
bsub -n 2 -m "host1 host2! hostgroupA! host3 host4" my_parallel_job
```

In this example, LSF runs `my_parallel_job` according to the following steps:

1. LSF selects either host2 or a host defined in `hostgroupA` as the first execution host for the parallel job.

   Note: First execution host candidates specified at the job-level (command line) override candidates defined at the queue level (in `lsb.queues`).
2. If any of the first execution host candidates have enough processors to run the job, the entire job runs on the first execution host, and not on any other hosts.

In the example, if host2 or a member of hostgroupA has two or more processors, the entire job runs on the first execution host.

3. If the first execution host does not have enough processors to run the entire job, LSF selects additional hosts that are not defined as first execution host candidates.

   Follow these guidelines when you specify first execution host candidates:

   - If you specify a host group, you must first define the host group in the file lsb.hosts.
   - Do not specify a dynamic host group as a first execution host.
   - Do not specify all, allremote, or others, or a host partition as a first execution host.
   - Do not specify a preference (+) for a host identified as a first execution host candidate (using !).
   - For each parallel job, specify enough regular hosts to satisfy the processor requirement for the job. Once LSF selects a first execution host for the current job, the other first execution host candidates become unavailable to the current job, but remain available to other jobs as either regular or first execution hosts.
   - You cannot specify first execution host candidates when you use the **brun** command.

When using the LSF multicluster capability, insert an exclamation point (!) after the cluster name, as shown in the following example:

```
bsub -n 2 -m "host2@cluster2! host3@cluster2" my_parallel_job
```

When specifying compute units, the job runs within the listed compute units. Used in conjunction with a mandatory first execution host, the compute unit containing the first execution host is given preference.

In the following example one host from host group hg appears first, followed by other hosts within the same compute unit. Remaining hosts from other compute units appear grouped by compute, and in the same order as configured in the ComputeUnit section of lsb.hosts.

```
bsub -n 64 -m "hg! cu1 cu2 cu3 cu4" -R "cu[pref=config]" my_job
```

Examples

```
bsub -m "host1 host3 host8 host9" my_program
```

Submits my_program to run on one of the candidate hosts: host1, host3, host8, or host9.

-mig

Specifies the migration threshold for checkpointable or rerunnable jobs, in minutes.

Categories

schedule

Synopsis

bsub -mig *migration_threshold*

Description

Enables automatic job migration and specifies the migration threshold, in minutes. A value of 0 (zero) specifies that a suspended job should be migrated immediately.

Command-level job migration threshold overrides application profile and queue-level settings.

Where a host migration threshold is also specified, and is lower than the job value, the host value is used.

-N

Sends the job report to you by mail when the job finishes.

Categories

notify

Synopsis

```
bsub -N
```

Description

When used without any other options, behaves the same as the default.

Use only with `-o`, `-oo`, `-I`, `-Ip`, and `-Is` options, which do not send mail, to force LSF to send you a mail message when the job is done.

Conflicting options

Do not use with the **bsub -Ne** option. The **bmod -Nn** command option cancels both the `-N` and `-Ne` mail notification options.

-Ne

Sends the job report to you by mail when the job exits (that is, when the job is in Exit status).

Categories

notify

Synopsis

```
bsub -Ne
```

Description

This option ensures that an email notification is only sent on a job error.

When used without any other options, behaves the same as the default.

Use only with `-o`, `-oo`, `-I`, `-Ip`, and `-Is` options, which do not send mail, to force LSF to send you a mail message when the job exits.

Conflicting options

Do not use with the **bsub -N** option. The **bmod -Nn** command option cancels both the `-N` and `-Ne` mail notification options.

-n

Submits a parallel job and specifies the number of tasks in the job.

Categories

resource

Synopsis

```
bsub -n min_tasks[,max_tasks]
```

Description

The number of tasks is used to allocate a number of slots for the job. Usually, the number of slots assigned to a job will equal the number of tasks specified. For example, one task will be allocated with one slot. (Some slots/processors may be on the same multiprocessor host).

You can specify a minimum and maximum number of tasks. For example, this job requests a minimum of 4, but can launch up to 6 tasks:

```
bsub -n 4,6 a.out
```

The job can start if at least the minimum number of slots/processors is available for the minimum number of tasks specified. If you do not specify a maximum number of tasks, the number you specify represents the exact number of tasks to launch.

If **PARALLEL_SCHED_BY_SLOT=Y** in `lsb.params`, this option specifies the number of slots required to run the job, not the number of processors.

When used with the `-R` option and a compound resource requirement, the number of slots in the compound resource requirement must be compatible with the minimum and maximum tasks specified.

Jobs that have fewer tasks than the minimum **TASKLIMIT** defined for the queue or application profile to which the job is submitted, or more tasks than the maximum **TASKLIMIT** are rejected. If the job has minimum and maximum tasks, the maximum tasks requested cannot be less than the minimum **TASKLIMIT**, and the minimum tasks requested cannot be more than the maximum **TASKLIMIT**.

For example, if the queue defines `TASKLIMIT=4 8`:

- **bsub -n 6** is accepted because it requests slots within the range of **TASKLIMIT**
- **bsub -n 9** is rejected because it requests more tasks than the **TASKLIMIT** allows
- **bsub -n 1** is rejected because it requests fewer tasks than the **TASKLIMIT** allows
- **bsub -n 6,10** is accepted because the minimum value 6 is within the range of the **TASKLIMIT** setting
- **bsub -n 1,6** is accepted because the maximum value 6 is within the range of the **TASKLIMIT** setting
- **bsub -n 10,16** is rejected because its range is outside the range of **TASKLIMIT**
- **bsub -n 1,3** is rejected because its range is outside the range of **TASKLIMIT**

See the **TASKLIMIT** parameter in `lsb.queues` and `lsb.applications` for more information.

If **JOB_SIZE_LIST** is defined in `lsb.applications` or `lsb.queues` and a job is submitted to a queue or an application profile with a job size list, the requested job size in the job submission must request only a single job size (number of tasks) rather than a minimum and maximum value and the requested job size in the job submission must satisfy the list defined in **JOB_SIZE_LIST**, otherwise LSF rejects the job submission. If a job submission does not include a job size request, LSF assigns the default job size to the submission request. **JOB_SIZE_LIST** overrides any **TASKLIMIT** parameters defined at the same level.

For example, if the application profile or queue defines `JOB_SIZE_LIST=4 2 10 6 8`:

- **bsub -n 6** is accepted because it requests a job size that is in **JOB_SIZE_LIST**.
- **bsub -n 9** is rejected because it requests a job size that is not in **JOB_SIZE_LIST**.
- **bsub -n 2,8** is rejected because you cannot request a range of job slot sizes when **JOB_SIZE_LIST** is defined.
- **bsub** without specifying a job size (using `-n` or `-R`) is accepted and the job submission is assigned a job size request of 4 (the default value).

See the **JOB_SIZE_LIST** parameter in `lsb.applications`(5) or `lsb.queues`(5) for more information.

In the LSF multicluster capability environment, if a queue exports jobs to remote clusters (see the **SNDJOBS_TO** parameter in `lsb.queues`), then the process limit is not imposed on jobs submitted to this queue.

Once the required number of processors is available, the job is dispatched to the first host selected. The list of selected host names for the job are specified in the environment variables **LSB_HOSTS** and

LSB_MCPU_HOSTS. The job itself is expected to start parallel components on these hosts and establish communication among them, optionally using RES.

Specify first execution host candidates using the `-m` option when you want to ensure that a host has the required resources or runtime environment to handle processes that run on the first execution host.

If you specify one or more first execution host candidates, LSF looks for a first execution host that satisfies the resource requirements. If the first execution host does not have enough processors or job slots to run the entire job, LSF looks for additional hosts.

Examples

```
bsub –n2 –R "span[ptile=1]" –network "protocol=mpi,lapi: type=sn_all:
instances=2: usage=shared" poe /home/user1/mpi_prog
```

For this job running on `hostA` and `hostB`, each task will reserve 8 windows (2*2*2), for 2 protocols, 2 instances and 2 networks. If enough network windows are available, other network jobs with `usage=shared` can run on `hostA` and `hostB` because networks used by this job are shared.

-notify

Requests that the user be notified when the job reaches any of the specified states.

Categories

notify

Synopsis

```
bsub -notify "[exit ][done ][start ][suspend]"
```

Description

Use a space to separate multiple job states. The notification request is logged in **JOB_NEW** events.

-network

For LSF IBM Parallel Environment (IBM PE) integration. Specifies the network resource requirements to enable network-aware scheduling for IBM PE jobs.

Categories

resource

Synopsis

`bsub -network "` *network_res_req* `"`

Description

If any network resource requirement is specified in the job, queue, or application profile, the jobs are treated as IBM PE jobs. IBM PE jobs can only run on hosts where IBM PE **pnsd** daemon is running.

The network resource requirement string *network_res_req* has the same syntax as the NETWORK_REQ parameter defined in `lsb.applications` or `lsb.queues`.

network_res_req has the following syntax:

[`type=sn_all` | `sn_single`] [`:protocol=`*protocol_name*[(*protocol_number*)] [`,`*protocol_name*[(*protocol_number*)]] [`:mode=US` | `IP`] [`:usage=shared` | `dedicated`] [`:instance=`*positive_integer*]

LSF_PE_NETWORK_NUM must be defined to a non-zero value in `lsf.conf` for the LSF to recognize the `-network` option. If **LSF_PE_NETWORK_NUM** is not defined or is set to 0, the job submission is rejected with a warning message.

The `-network` option overrides the value of NETWORK_REQ defined in `lsb.applications` or `lsb.queues`.

The following network resource requirement options are supported:

type=sn_all | sn_single
Specifies the adapter device type to use for message passing: either `sn_all` or `sn_single`.

`sn_single`
When used for switch adapters, specifies that all windows are on a single network

`sn_all`
Specifies that one or more windows are on each network, and that striped communication should be used over all available switch networks. The networks specified must be accessible by all hosts selected to run the IBM PE job. See the *Parallel Environment Runtime Edition for AIX: Operation and Use* guide (SC23-6781-05) for more information about submitting jobs that use striping.

If `mode` is `IP` and type is specified as `sn_all` or `sn_single`, the job will only run on IB adapters (IPoIB). If mode is `IP` and `type` is not specified, the job will only run on Ethernet adapters (IPoEth). For IPoEth jobs, LSF ensures the job is running on hosts where **pnsd** is installed and running. For IPoIB jobs, LSF ensures the job the job is running on hosts where **pnsd** is installed and running, and that InfiniBand networks are up. Because IP jobs do not consume network windows, LSF does not check if all network windows are used up or the network is already occupied by a dedicated IBM PE job.

Equivalent to the IBM PE MP_EUIDEVICE environment variable and `-euidevice` IBM PE flag See the *Parallel Environment Runtime Edition for AIX: Operation and Use* guide (SC23-6781-05) for more information. Only `sn_all` or `sn_single` are supported by LSF. The other types supported by IBM PE are not supported for LSF jobs.

protocol=*protocol_name*[(*protocol_number*)]
Network communication protocol for the IBM PE job, indicating which message passing API is being used by the application. The following protocols are supported by LSF:

mpi
The application makes only MPI calls. This value applies to any MPI job regardless of the library that it was compiled with (IBM PE MPI, MPICH2).

pami
The application makes only PAMI calls.

lapi
The application makes only LAPI calls.

shmem
The application makes only OpenSHMEM calls.

user_defined_parallel_api
The application makes only calls from a parallel API that you define. For example: `protocol=myAPI` or `protocol=charm`.

The default value is `mpi`.

LSF also supports an optional *protocol_number* (for example, `mpi(2)`, which specifies the number of contexts (endpoints) per parallel API instance. The number must be a power of 2, but no greater than 128 (1, 2, 4, 8, 16, 32, 64, 128). LSF will pass the communication protocols to IBM PE without any change. LSF will reserve network windows for each protocol.

When you specify multiple parallel API protocols, you cannot make calls to both LAPI and PAMI (`lapi`, `pami`) or LAPI and OpenSHMEM (`lapi`, `shmem`) in the same application. Protocols can be specified in any order.

See the MP_MSG_API and MP_ENDPOINTS environment variables and the `-msg_api` and `-endpoints` IBM PE flags in the *Parallel Environment Runtime Edition for AIX: Operation and Use* guide (SC23-6781-05) for more information about the communication protocols that are supported by IBM PE.

mode=US | IP
The network communication system mode used by the communication specified communication protocol: US (User Space) or IP (Internet Protocol). The default value is US. A US job can only run with adapters that support user space communications, such as the IB adapter. IP jobs can run with either Ethernet adapters or IB adapters. When IP mode is specified, the instance number cannot be specified, and network usage must be unspecified or shared.

Each instance on the US mode requested by a task running on switch adapters requires and adapter window. For example, if a task requests both the MPI and LAPI protocols such that both protocol instances require US mode, two adapter windows will be used.

usage=dedicated | shared
Specifies whether the adapter can be shared with tasks of other job steps: `dedicated` or `shared`. Multiple tasks of the same job can share one network even if usage is `dedicated`.

instance=*positive_integer*
The number of parallel communication paths (windows) per task made available to the protocol on each network. The number actually used depends on the implementation of the protocol subsystem.

The default value is `1`.

If the specified value is greater than MAX_PROTOCOL_INSTANCES in `lsb.params` or `lsb.queues`, LSF rejects the job.

The following IBM LoadLeveller job command file options are not supported in LSF:

- `collective_groups`
- `imm_send_buffers`
- `rcxtblocks`

See *Administering IBM Spectrum LSF* for more information about network-aware scheduling and running and managing workload through IBM Parallel Environment.

Examples

```
bsub –n2 –R "span[ptile=1]" –network "protocol=mpi,lapi: type=sn_all:
instances=2: usage=shared" poe /home/user1/mpi_prog
```

For this job running on `hostA` and `hostB`, each task will reserve 8 windows (2*2*2), for 2 protocols, 2 instances and 2 networks. If enough network windows are available, other network jobs with `usage=shared` can run on `hostA` and `hostB` because networks used by this job are shared.

-nnodes

Specifies the number of compute nodes that are required for the CSM job.

Categories

resource

Synopsis

`bsub -nnodes` *num_nodes*

Conflicting options

Do not use with the `-csm y` option.

Description

This option is for easy mode LSF job submission to run with the IBM Cluster Systems Manager (CSM) integration package.

Default

1

-o

Appends the standard output of the job to the specified file path.

Categories

io

Synopsis

`bsub -o` *output_file*

Description

Sends the output by mail if the file does not exist, or the system has trouble writing to it.

If only a file name is specified, LSF writes the output file to the current working directory. If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard output file to `/tmp/`.

If the specified *output_file* path is not accessible, the output will not be stored.

If you use the special character `%J` in the name of the output file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the output file, then `%I` is replaced by the index of the job in the array, if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

If the parameter **LSB_STDOUT_DIRECT** in `lsf.conf` is set to Y or y, the standard output of a job is written to the file you specify as the job runs. If **LSB_STDOUT_DIRECT** is not set, it is written to a temporary file and copied to the specified file after the job finishes. **LSB_STDOUT_DIRECT** is not supported on Windows.

If you use `-o` without `-e` or `-eo`, the standard error of the job is stored in the output file.

If you use `-o` with `-e` or `-eo` and the specified *error_file* is the same as the *output_file*, the *error_file* of the job is NULL and the standard error of the job is stored in the output file.

If you use `-o` without `-N`, the job report is stored in the output file as the file header.

If you use both `-o` and `-N`, the output is stored in the output file and the job report is sent by mail. The job report itself does not contain the output, but the report advises you where to find your output.

Examples

```
bsub -q short -o my_output_file "pwd; ls"
```

Submit the UNIX command **pwd** and **ls** as a job to the queue named `short` and store the job output in `my_output` file.

-oo

Overwrites the standard output of the job to the specified file path.

Categories

io

Synopsis

`bsub -oo` *output_file*

Description

Overwrites the standard output of the job to the specified file if it exists, or sends the output to a new file if it does not exist. Sends the output by mail if the system has trouble writing to the file.

If only a file name is specified, LSF writes the output file to the current working directory. If the current working directory is not accessible on the execution host after the job starts, LSF writes the standard output file to `/tmp/`.

If the specified *output_file* path is not accessible, the output will not be stored.

If you use the special character `%J` in the name of the output file, then `%J` is replaced by the job ID of the job. If you use the special character `%I` in the name of the output file, then `%I` is replaced by the index of the job in the array, if the job is a member of an array. Otherwise, `%I` is replaced by 0 (zero).

Note: The file path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

If the parameter **LSB_STDOUT_DIRECT** in `lsf.conf` is set to Y or y, the standard output of a job overwrites the output file you specify as the job runs, which occurs every time the job is submitted with the overwrite option, even if it is requeued manually or by the system. If **LSB_STDOUT_DIRECT** is not set, the output is written to a temporary file that overwrites the specified file after the job finishes. **LSB_STDOUT_DIRECT** is not supported on Windows.

If you use `-oo` without `-e` or `-eo`, the standard error of the job is stored in the output file.

If you use `-oo` without `-N`, the job report is stored in the output file as the file header.

If you use both `-oo` and `-N`, the output is stored in the output file and the job report is sent by mail. The job report itself does not contain the output, but the report advises you where to find your output.

-outdir

Creates the job output directory.

Categories

io

Synopsis

`bsub -outdir` *output_directory*

Description

The path for the output directory can include the following dynamic patterns, which are case sensitive:

- %J - job ID
- %JG - job group (if not specified, it will be ignored)
- %I - index (default value is 0)
- %EJ - execution job ID
- %EI - execution index
- %P - project name
- %U - user name
- %G - User group new forthe job output directory
- %H - first execution host name

For example, the system creates the `submission_dir/user1/jobid_0/` output directory for the job with the following command:

```
bsub -outdir "%U/%J_%I" myprog
```

If the cluster wide output directory was defined but the outdir option was not set, for example, **DEFAULT_JOB_OUTDIR**=/scratch/joboutdir/%U/%J_%I in lsb.params, the system creates the /scratch/joboutdir/user1/jobid_0/ output directory for the job with the following command:

```
bsub myprog
```

If the submission directory is /scratch/joboutdir/ on the shared file system and you want the system to create /scratch/joboutdir/user1/jobid_0/ for the job output directory, then run the following command:

```
bsub -outdir "%U/%J_%I" myjob
```

Since the command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name, the outdir option does not have its own length limitation. The outdir option supports mixed UNIX and Windows paths when **LSB_MIXED_PATH_ENABLE=Y/y**. **LSB_MIXED_PATH_DELIMITER** controls the delimiter.

The following assumptions and dependencies apply to the -outdir command option:

- The execution host has access to the submission host.
- The submission host should be running RES or it will use EGO_RSH to run a directory creation command. If this parameter is not defined, rsh will be used. RES should be running on the Windows submission host in order to create the output directory.

-P

Assigns the job to the specified project.

Categories

properties

Synopsis

bsub -P *project_name*

Description

The project does not have to exist before submitting the job.

Project names can be up to 511 characters long.

-p

Sets the limit of the number of processes to the specified value for the whole job.

Categories

limit

Synopsis

bsub -p *process_limit*

Description

The default is no limit. Exceeding the limit causes the job to terminate.

-pack

Submits job packs instead of an individual job.

Categories

pack

Synopsis

`bsub -pack` *job_submission_file*

Conflicting options

Do not use with any other **bsub** option in the command line.

Description

The purpose of the job packs feature is to speed up the submission of a large number of jobs. When job pack submission is enabled, you can submit jobs by submitting a single file containing multiple job requests.

Specify the full path to the job submission file. The job packs feature must be enabled (by defining **LSB_MAX_PACK_JOBS** in `lsb.conf`) to use the `-pack` option.

In the command line, this option is not compatible with any other **bsub** options. Do not put any other **bsub** options in the command line, as they must be included in each individual job request in the file.

In the job submission file, define one job request per line, using normal **bsub** syntax but omitting the word "bsub". For requests in the file, job pack submission supports all **bsub** options in the job submission file except for the following:

`-I`, `-Ip`, `-Is`, `-IS`, `-ISp`, `-ISs`, `-IX`, `-XF`, `-K`, `-jsdl`, `-h`, `-V`, `-pack`.

When you use the job packs feature to submit multiple jobs to mbatchd at once, instead of submitting the jobs individually, it minimizes system overhead and improves the overall job submission rate dramatically. When you use this feature, you create a job submission file that defines each job request. You specify all the **bsub** options individually for each job, so unlike chunk jobs and job arrays, there is no need for jobs in this file to have anything in common. To submit the jobs to LSF, you simply submit the file using the **bsub -pack** option.

LSF parses the file contents and submits the job requests to mbatchd, sending multiple requests at one time. Each group of jobs submitted to mbatchd together is called a job pack. The job submission file can contain any number of job requests, and LSF will group them into job packs automatically. The reason to group jobs into packs is to maintain proper mbatchd performance: while mbatchd is processing a job pack, mbatchd is blocked from processing other requests, so limiting the number of jobs in each pack ensures a reasonable mbatchd response time for other job submissions. Job pack size is configurable.

If the cluster configuration is not consistent and **mbatchd** receives a job pack that exceeds the job pack size defined in `lsf.conf`, it will be rejected.

Once the pack is submitted to mbatchd, each job request in the pack is handled by LSF as if it was submitted individually with the **bsub** command.

Enabling job packs

About this task

Job packs are disabled by default. Before running **bsub -pack**, you must enable the job packs feature.

Procedure

1. Edit `lsf.conf`.
2. Define the parameter `LSB_MAX_PACK_JOBS=100`.

   Defining this parameter enables the job packs feature and sets the job pack size. Set 100 as the initial pack size and modify the value as needed. If set to 1, jobs from the file are submitted individually, as if submitted directly using the **bsub** command. If set to 0, job packs are disabled.
3. Optionally, define the parameter `LSB_PACK_MESUB=N`.

   Do this if you want to further increase the job submission rate by preventing the execution of any **mesub** during job submission. This parameter only affects the jobs submitted using job packs.

4. Optionally, define the parameter LSB_PACK_SKIP_ERROR=Y.

   Do this if you want LSF to process all requests in a job submission file, and continue even if some requests have errors.
5. Restart **mbatchd** to make your changes take effect.

Submitting job packs

Procedure

1. Prepare the job submission file.

   The job submission file is a text file containing all the jobs that you want to submit. Each line in the file is one job request. For each request, the syntax is identical to the **bsub** command line without the word "bsub".

   For example,

   ```
   #This file contains 2 job requests.
   -R "select[mem>200] rusage[mem=100]" job1.sh
   -R "select[swap>400] rusage[swap=200]" job2.sh
   #end
   ```

   The job submission file has the following limitations:

   - The following **bsub** options are not supported:

     -I, -Ip, -Is, -IS,-ISp, -ISs, -IX, -XF, -K, -jsdl, -h, -V, -pack.
   - Terminal Services jobs are not supported.
   - I/O redirection is not supported.
   - Blank lines and comment lines (beginning with #) are ignored. Comments at the end of a line are not supported.
   - Backslash (\) is not considered a special character to join two lines.
   - Shell scripting characters are treated as plain text, they will not be interpreted.
   - Matched pairs of single and double quotations are supported, but they must have space before and after. For example, **-J "job1"** is supported, **-J"job1"** is not, and **-J "job"1** is not.

   For job dependencies, use the job name instead of job ID to specify the dependency condition. A job request will be rejected if the job name or job ID of the job it depends on does not already exist.
2. After preparing the file, use the **bsub -pack** option to submit all the jobs in the file.

   Specify the full path to the job submission file. Do not put any other **bsub** options in the command line, they must be included in each individual job request in the file.

-ptl

Specifies the pending time limit for the job.

Categories

limit

Synopsis

bsub -ptl [*hour*:]*minute*

Description

LSF sends the job-level pending time limit configuration to IBM Spectrum LSF RTM (LSF RTM), which handles the alarm and triggered actions such as user notification (for example, notifying the user that submitted the job and the LSF administrator) and job control actions (for example, killing the job). LSF RTM compares the job's pending time to the pending time limit, and if the job is pending for longer than

this specified time limit, LSF RTM triggers the alarm and actions. This parameter works without LSF RTM, but LSF does not take any other alarm actions.

In MultiCluster job forwarding mode, the job's pending time limit is ignored in the execution cluster, while the submission cluster merges the job's queue-, application-, and job-level pending time limit according to local settings.

The pending time limit is in the form of [*hour*:]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The job-level pending time limit specified here overrides any application-level or queue-level limits specified (**PEND_TIME_LIMIT** in `lsb.applications` and `lsb.queues`).

-Q

Specify automatic job requeue exit values.

Categories

properties

Synopsis

`bsub -Q "`*exit_code* [*exit_code* ...] [EXCLUDE(*exit_code* ...)]"

Description

Use spaces to separate multiple exit codes. The reserved keyword all specifies all exit codes. Exit codes are typically between 0 and 255. Use a tilde (~) to exclude specified number or numbers from the list.

exit_code has the following form:

```
"[all] [~number ...] | [number ...]"
```

Job level exit values override application-level and queue-level values.

Jobs running with the specified exit code share the same application and queue with other jobs.

Define an exit code as EXCLUDE(*exit_code*) to enable exclusive job requeue. Exclusive job requeue does not work for parallel jobs.

If **mbatchd** is restarted, it does not remember the previous hosts from which the job exited with an exclusive requeue exit code. In this situation, it is possible for a job to be dispatched to hosts on which the job has previously exited with an exclusive exit code.

-q

Submits the job to one of the specified queues.

Categories

properties

Synopsis

`bsub -q "`*queue_name* ..."

Description

Quotes are optional for a single queue. The specified queues must be defined for the local cluster. For a list of available queues in your local cluster, use **bqueues**.

When a list of queue names is specified, LSF attempts to submit the job to the first queue listed. If that queue cannot be used because of the job's resource limits or other restrictions, such as the requested hosts, your accessibility to a queue, queue status (closed or open), then the next queue listed is

considered. The order in which the queues are considered is the same order in which these queues are listed.

Examples

```
bsub -q short -o my_output_file "pwd; ls"
```

Submit the UNIX command **pwd** and **ls** as a job to the queue named `short` and store the job output in `my_output` file.

```
bsub -q "queue1 queue2 queue3" -c 5 my_program
```

Submit `my_program` to one of the candidate queues: `queue1`, `queue2`, and `queue3` that are selected according to the CPU time limit specified by **-c 5**.

-R

Runs the job on a host that meets the specified resource requirements.

Categories

resource

Synopsis

`bsub -R "`*res_req*`" [-R "`*res_req*`" ...]`

Description

A resource requirement string describes the resources a job needs. LSF uses resource requirements to select hosts for job execution. Resource requirement strings can be simple (applying to the entire job), compound (applying to the specified number of slots), or alternative.

Simple resource requirement strings are divided into the following sections. Each section has a different syntax.

- A selection section (`select`). The selection section specifies the criteria for selecting execution hosts from the system.
- An ordering section (`order`). The ordering section indicates how the hosts that meet the selection criteria should be sorted.
- A resource usage section (`rusage`). The resource usage section specifies the expected resource consumption of the task.
- A job spanning section (`span`). The job spanning section indicates if a parallel job should span across multiple hosts.
- A same resource section (`same`). The same section indicates that all processes of a parallel job must run on the same type of host.
- A compute unit resource section (`cu`). The compute unit section specifies topological requirements for spreading a job over the cluster.
- A CPU and memory affinity resource section (`affinity`). The affinity section specifies CPU and memory binding requirements for tasks of a job.

The resource requirement string sections have the following syntax:

```
select[selection_string] order[order_string] rusage[
usage_string [, usage_string][|| usage_string] ...]
span[span_string] same[same_string] cu[cu_string]] affinity[affinity_string]
```

The square brackets must be typed as shown for each section. A blank space must separate each resource requirement section.

For example, to submit a job that runs on Red Hat Enterprise Linux 6 or Red Hat Enterprise Linux 7:

```
bsub -R "rhel6 || rhel7" myjob
```

The following command runs the job called `myjob` on an HP-UX host that is lightly loaded (CPU utilization) and has at least 15 MB of swap memory available.

```
bsub -R "swp > 15 && hpux order[ut]" myjob
```

You can omit the `select` keyword (and its square brackets), but if you include a `select` section, it must be the first string in the resource requirement string. If you do not give a section name, the first resource requirement string is treated as a selection string (`select[`*`selection_string`*`]`).

For example, the following resource requirements are equivalent:

```
bsub -R "type==any order[ut] same[model] rusage[mem=1]" myjob
```

```
bsub -R "select[type==any] order[ut] same[model] rusage[mem=1]" myjob
```

If you need to include a hyphen (`-`) or other non-alphabetic characters within the string, enclose the text in single quotation marks, for example, **`bsub -R "select[hname!='host06-x12']"`**.

If the LSF_STRICT_RESREQ=Y parameter is specified in the `lsf.conf` file, the selection string must conform to the stricter resource requirement string syntax described in *Administering IBM Spectrum LSF*. The strict resource requirement syntax only applies to the select section. It does not apply to the other resource requirement sections (order, rusage, same, span, or cu). When LSF_STRICT_RESREQ=Y in `lsf.conf`, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

If **RESRSV_LIMIT** is set in `lsb.queues`, the merged application-level and job-level rusage consumable resource requirements must satisfy any limits set by **RESRSV_LIMIT**, or the job will be rejected.

Any resource for run queue length, such as r15s, r1m or r15m, specified in the resource requirements refers to the normalized run queue length.

By default, memory (`mem`) and swap (`swp`) limits in `select[]` and `rusage[]` sections are specified in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for these limits (MB, GB, TB, PB, EB, or ZB).

You can use the following units for resource requirements and limits:

- KB or K (kilobytes)
- MB or M (megabytes)
- GB or G (gigabytes)
- TB or T (terabytes)
- PB or P (petabytes)
- EB or E (exabytes)
- ZB or Z (zettabytes)

The specified unit is converted to the appropriate value specified by the **LSF_UNIT_FOR_LIMITS** parameter. The converted limit values round up to a positive integer. For resource requirements, you can specify unit for `mem`, `swp` and `tmp` in `select` and `rusage` section.

By default, the `tmp` resource is not supported by the **LSF_UNIT_FOR_LIMITS** parameter. Use the parameter **LSF_ENABLE_TMP_UNIT=Y** to enable the **LSF_UNIT_FOR_LIMITS** parameter to support limits on the `tmp` resource.

If the **LSF_ENABLE_TMP_UNIT=Y** and **LSF_UNIT_FOR_LIMIT=GB** parameters are set, the following conversion happens.

```
bsub -C 500MB -M 1G -S 1TB -F 1M -R "rusage[mem=512MB:swp=1GB:tmp=1TB]" sleep 100
```

The units in this job submission are converted to the following units:

```
bsub -C 1 -M 1 -S 1024 -F 1024 -R "rusage[mem=0.5:swp=1:tmp=1024]" sleep 100
```

Multiple resource requirement strings

The **bsub** command also accepts multiple -R options for the `order`, `same`, `rusage` (not multi-phase), and `select` sections. You can specify multiple strings instead of using the && operator:

```
bsub -R "select[swp > 15]" -R "select[hpux] order[r15m]"
-R rusage[mem=100]" -R "order[ut]" -R "same[type]"
-R "rusage[tmp=50:duration=60]" -R "same[model]" myjob
```

LSF merges the multiple -R options into one string and selects a host that meets all of the resource requirements. The number of -R option sections is unlimited.

Remember: Use multiple -R options only with the `order`, `same`, `rusage` (not multi-phase), and `select` sections of simple resource requirement strings and with the **bsub** and **bmod** commands.

When application-level and queue-level `cu` sections are also defined, the job-level `cu` section takes precedence and overwrites the application-level requirement definition, which in turn takes precedence and overwrites the queue-level requirement definitions.

For example, when `EXCLUSIVE=CU[enclosure]` is specified in the `lsb.queues` file, with a compute unit type `enclosure` in the `lsf.params` file, and `ComputeUnit` section in the `lsb.hosts` file, the following command submits a job that runs on 64 slots over 4 enclosures or less, and uses the enclosures exclusively:

```
bsub -n 64 -R "cu[excl:type=enclosure:maxcus=4]" myjob
```

A resource called `bigmem` is defined in the `lsf.shared` file as an exclusive resource for host `hostE` in the `lsf.cluster` file. Use the following command to submit a job that runs on `hostE`:

```
bsub -R "bigmem" myjob
```

or

```
bsub -R "defined(bigmem)" myjob
```

A static shared resource is configured for licenses for the Verilog application as a resource called `verilog_lic`. To submit a job that runs on a host when there is a license available:

```
bsub -R "select[defined(verilog_lic)] rusage[verilog_lic=1]" myjob
```

The following job requests 20 MB memory for the duration of the job, and 1 license for 2 minutes:

```
bsub -R "rusage[mem=20, license=1:duration=2]" myjob
```

The following job requests 20 MB of memory, 50 MB of swap space for 1 hour, and 1 license for 2 minutes:

```
bsub -R "rusage[mem=20:swp=50:duration=1h, license=1:duration=2]" myjob
```

The following job requests 20 MB of memory for the duration of the job, 50 MB of swap space for 1 hour, and 1 license for 2 minutes.

```
bsub -R "rusage[mem=20,swp=50:duration=1h, license=1:duration=2]" myjob
```

The following job requests 50 MB of swap space, linearly decreasing the amount reserved over a duration of 2 hours, and requests 1 license for 2 minutes:

```
bsub -R "rusage[swp=50:duration=2h:decay=1, license=1:duration=2]" myjob
```

The following job requests two resources with same duration but different decay:

```
bsub -R "rusage[mem=20:duration=30:decay=1, lic=1:duration=30]" myjob
```

The following job uses a multi-phase `rusage` string to request 50 MB of memory for 10 minutes, followed by 10 MB of memory for the duration of the job:

```
bsub -R "rusage[mem=(50 10):duration=(10):decay=(0)]" myjob
```

In the following example, you are running an application version 1.5 as a resource called `app_lic_v15` and the same application version 2.0.1 as a resource called `app_lic_v201`. The license key for version 2.0.1 is backward compatible with version 1.5, but the license key for version 1.5 does not work with 2.0.1.

Note: Job-level resource requirement specifications that use the OR (`||`) operator take precedence over any queue-level resource requirement specifications.

- If you can only run your job using one version of the application, submit the job without specifying an alternative resource. To submit a job that only uses `app_lic_v201`:

  ```
  bsub -R "rusage[app_lic_v201=1]" myjob
  ```

- If you can run your job using either version of the application, try to reserve version 2.0.1 of the application. If it is not available, you can use version 1.5. To submit a job that tries `app_lic_v201` before trying `app_lic_v15`:

  ```
  bsub -R "rusage[app_lic_v201=1||app_lic_v15=1]" myjob
  ```

- If different versions of an application require different system resources, you can specify other resources in your `rusage` strings. To submit a job that uses 20 MB of memory for `app_lic_v201` or 20 MB of memory and 50 MB of swap space for `app_lic_v15`:

  ```
  bsub -R "rusage[mem=20:app_lic_v15=1||mem=20:swp=50:app_lic_v201=1]" myjob
  ```

You can specify a threshold at which the consumed resource must be at before an allocation should be made. For example,

```
bsub -R "rusage[bwidth=1:threshold=5]" myjob
```

For example, a job is submitted that consumes 1 unit of bandwidth (the resource `bwidth`), but the job should not be scheduled to run unless the bandwidth on the host is equal to or greater than 5. In this example, `bwidth` is a decreasing resource and the threshold value is interpreted as a floor. If the resource in question was increasing, then the threshold value would be interpreted as a ceiling.

An *affinity resource requirement* string specifies CPU and memory binding requirements for a resource allocation that is topology aware. An `affinity[]` resource requirement section controls the allocation and distribution of *processor units* within a host according to the hardware topology information that LSF collects.

Resource reservation method

Specify the resource reservation method in the resource usage string by using the **/job**, **/host**, or **/task** keyword after the numeric value. The resource reservation method specified in the resource string overrides the global setting that is specified in the **ReservationUsage** section of the `lsb.resources` file. You can only specify resource reservation methods for consumable resources. Specify the resource reservation methods as follows:

- *value***/job**

  Specifies per-job reservation of the specified resource. This is the equivalent of specifying PER_JOB for the **METHOD** parameter in the **ReservationUsage** section of the `lsb.resources` file.

- *value***/host**

  Specifies per-host reservation of the specified resource. This is the equivalent of specifying PER_HOST for the **METHOD** parameter in the **ReservationUsage** section of the `lsb.resources` file.

- *value*`/task`

  Specifies per-task reservation of the specified resource. This is the equivalent of specifying PER_TASK for the **METHOD** parameter in the **ReservationUsage** section of the `lsb.resources` file.

- Basic syntax:

  ```
  resource_name=value/method:duration=value:decay=value
  ```

  For example,

  ```
  rusage[mem=10/host:duration=10:decay=0]
  ```

- Multi-phase memory syntax:

  ```
  resource_name=(value ...)/method:duration=(value ...):decay=value
  ```

  For example,

  ```
  rusage[mem=(50 20)/task:duration=(10 5):decay=0]
  ```

Compound resource requirements

In some cases different resource requirements may apply to different parts of a parallel job. The first execution host, for example, may require more memory or a faster processor for optimal job scheduling. *Compound resource requirements* allow you to specify different requirements for some slots within a job in the queue-level, application-level, or job-level resource requirement string.

Compound resource requirement strings can be set by the application-level or queue-level **RES_REQ** parameter, or used with the **bsub -R** option when a job is submitted. The **bmod -R** option accepts compound resource requirement strings for pending jobs but not running jobs.

Special rules take effect when compound resource requirements are merged with resource requirements defined at more than one level. If a compound resource requirement is used at any level (job, application, or queue) the compound multi-level resource requirement combinations apply.

Compound resource requirement strings are made up of one or more simple resource requirement strings as follows:

```
num1*{simple_string1} + num2*{simple_string2} + ...
```

where *numx* is the number of slots affected and *simple_stringx* is a simple resource requirement string.

The same resource requirement can be used within each component expression (simple resource requirement). For example, for static string resource `res1` and `res2`, a resource requirement such as the following is permitted:

"4*{select[io] same[res1]} + 4*{select[compute] same[res1]}"

With this resource requirement, there are two simple subexpressions, R1 and R2. For each of these subexpressions, all slots must come from hosts with equal values of res1. However, R1 may occupy hosts of a different value than those occupied by R2.

You can specify a global same requirement that takes effect over multiple subexpressions of a compound resource requirement string. For example,

```
"{4*{select[io]} + 4*{select[compute]}} same[res1]"
```

This syntax allows users to express that both subexpressions must reside on hosts that have a common value for `res1`.

In general, there may be more than two subexpressions in a compound resource requirement. The global same will apply to all of them.

Arbitrary nesting of brackets is not permitted. For example, you cannot have a global same apply to only two of three subexpressions of a compound resource requirement. However, each subexpression can have its own local same as well as a global same for the compound expression as a whole. For example, the following is permitted:

```
"{4*{same[res1]} + 4*{same[res1]}} same[res2]"
```

In addition, a compound resource requirement expression with a global same may be part of a larger alternative resource requirement string.

A compound resource requirement expression with a global same can be used in the following instances:

- Submitting a job: **bsub -R "res_req_string" <other_bsub_options> a.out**
- Configuring application profile (`lsb.applications` file): **RES_REQ = "res_req_string"**
- Queue configuration (`lsb.queues` file): **RES_REQ = "res_req_string"**

Syntax:

- A single compound resource requirement:

  "{*compound_res_req*} same[*same_string*]"
- A compound resource requirement within an alternative resource requirement:

  "{{*compound_res_req*} same[*same_string*]} || {*alt_res_req*}"
- A compound resource requirement within an alternative resource requirement with delay:

  "{*alt_res_req*} || {{*compound_res_req*} same[*same_string*]}@*delay*"

  The *delay* option is a positive integer.

Restriction:

- Compound resource requirements cannot contain the || operator. Compound resource requirements cannot be defined (included) in any multiple -R options.
- Compound resource requirements cannot contain the compute unit (cu) keywords `balance` or `excl`, but works normally with other `cu` keywords (including `pref`, `type`, `maxcus`, and `usablecuslots`).
- Resizable jobs can have compound resource requirements, but only the portion of the job represented by the last term of the compound resource requirement is eligible for automatic resizing. When you use the **bresize release** to release slots, you can release only slots represented by the last term of the compound resource requirement. To release slots in earlier terms, run the **bresize release** command repeatedly to release slots in subsequent last terms.
- Compound resource requirements cannot be specified in the definition of a guaranteed resource pool.
- Resource allocation for parallel jobs using compound resources is done for each compound resource term in the order listed instead of considering all possible combinations. A host rejected for not satisfying one resource requirement term will not be reconsidered for subsequent resource requirement terms.

For jobs without the number of total slots specified with the **bsub -n** command, the final *numx* can be omitted. The final resource requirement is then applied to the zero or more slots not yet accounted for:

- (*final res_req number of slots*) = (*total number of job slots*)-(*num1*+*num2*+ ...)

For jobs with the total number of slots specified with the **bsub -n** *num_slots* command, the total number of slots must match the number of slots in the resource requirement:

- *num_slots*=(*num1*+*num2*+*num3*+ ...)

For jobs with the minimum and maximum number of slots specified with the **bsub -n** *min, max* command, the number of slots in the compound resource requirement must be compatible with the minimum and maximum specified.

You can specify the number of slots or processors through the resource requirement specification. For example, you can specify a job that requests 10 slots or processors: 1 on a host that has more than 5000 MB of memory, and an additional 9 on hosts that have more than 1000 MB of memory:

```
bsub -R "1*{mem>5000} + 9*{mem>1000}" a.out
```

To specify compound GPU resource requirements, use the following keywords and options:

- In the `rusage[]` section, use the `ngpus_physical` resource to request the number of physical GPUs, together with the `gmodel` option specify the GPU model, the `gmem` option to specify the amount of reserved GPU memory, and the `nvlink` option to request GPUs with NVLink connections.
- In the `span[]` section, use the `gtile` keyword to specify the number of GPUs requested on each socket.

Note: If you specify GPU resource requirements with the **bsub -R** command option, or the **RES_REQ** parameter, the LSF ignores the `-gpu` option, as well as the **LSB_GPU_REQ** and **GPU_REQ** parameters.

The following example requests 2 hosts, reserving 2 GPUs, spread evenly across each socket for the first host, and 4 other hosts with 1 GPU on each host with 10 GB memory reserved for each GPU:

```
bsub -R "1*{span[gtile=!] rusage[ngpus_physical=2:gmem=1G]} + 4*{span[ptile=1]
rusage[ngpus_physical=1:gmem=10G]}" ./app
```

Alternative resource requirements

In some circumstances more than one set of resource requirements may be acceptable for a job to be able to run. LSF provides the ability to specify alternative resource requirements.

An alternative resource requirement consists of two or more individual simple or compound resource requirements. Each separate resource requirement describes an alternative. When a job is submitted with alternative resource requirements, the alternative resource picked must satisfy the mandatory first execution host. If none of the alternatives can satisfy the mandatory first execution host, the job will PEND.

Alternative resource requirement strings can be specified at the application-level or queue-level **RES_REQ** parameter, or used with **bsub -R** when a job is submitted. **bmod -R** also accepts alternative resource requirement strings for pending jobs.

The rules for merging job, application, and queue alternative resource requirements are the same as for compound resource requirements.

Alternative resource requirements cannot be used with the following features:

- Multiple **bsub -R** commands
- Taskstarter jobs, including those with the **tssub** command
- Hosts from HPC integrations that use toplib, including cpuset and Blue Gene hosts.
- Compute unit (cu) sections specified with `balance` or `excl` keywords.

If a job with alternative resource requirements specified is re-queued, it will have all alternative resource requirements considered during scheduling. If a `@D` delay time is specified, it is interpreted as waiting, starting from the original submission time. For a restart job, `@D` delay time starts from the restart job submission time.

An alternative resource requirement consists of two or more individual resource requirements. Each separate resource requirement describes an alternative. If the resources cannot be found that satisfy the first resource requirement, then the next resource requirement is tried, and so on until the requirement is satisfied.

Alternative resource requirements are defined in terms of a compound resource requirement, or an atomic resource requirement:

```
bsub -R "{C1 | R1 } || {C2 | R2 }@D2 || ... || {Cn | Rn }@Dn"
```

Where

- The OR operator (| |) separates one alternative resource from the next.
- The *C* option is a compound resource requirement.
- The *R* option is a resource requirement which is the same as the current LSF resource requirement, except when there is:
  - No rusage OR (| |).
  - No compute unit requirement `cu[...]`
- The *D* option is a positive integer:
  - @*D* is optional: Do not evaluate the alternative resource requirement until *D* minutes after submission time, and requeued jobs still use submission time instead of requeue time. There is no D1 because the first alternative is always evaluated immediately.
  - D2 <= D3 <= ... <= D*n*
  - Not specifying @*D* means that the alternative will be evaluated without delay if the previous alternative could not be used to obtain a job's allocation.

For example, you may have a sequential job, but you want alternative resource requirements (that is, if LSF fails to match your resource, try another one).

```
bsub -R "{ select[type==any] order[ut] same[model] rusage[mem=1] } ||
{ select[type==any] order[ls] same[ostype] rusage[mem=5] }" myjob
```

You can also add a delay before trying the second alternative:

```
bsub -R "{ select[type==any] order[ut] same[model] rusage[mem=1] } ||
{ select[type==any] order[ls] same[ostype] rusage[mem=5] }@4" myjob
```

You can also have more than 2 alternatives:

```
bsub -R "{select[type==any] order[ut] same[model] rusage[mem=1] } ||
{ select[type==any] order[ut] same[model] rusage[mem=1] } ||
{ select[type==any] order[ut] same[model] rusage[mem=1] }@3 ||
{ select[type==any] order[ut] same[model] rusage[mem=1] }@6" myjob
```

Some parallel jobs might need compound resource requirements. You can specify alternatives for parallel jobs the same way. That is, you can have several alternative sections each with brace brackets ({ }) around them separated by | |):

```
bsub -n 2 -R "{ 1*{ select[type==any] order[ut] same[model] rusage[mem=1]} + 1
*{ select[type==any] order[ut] same[model] rusage[mem=1] } } ||
{  1*{ select[type==any] order[ut] same[model] rusage[mem=1]} +
1*{ select[type==any] order[ut] same[model]
rusage[mem=1] } }@6" myjob
```

Alternatively, the compound resource requirement section can have both slots requiring the same resource:

```
bsub -n 2 -R "{ 1*{ select[type==any] order[ut] same[model] rusage[mem=1]}
+1*{ select[type==any] order[ut] same[model] rusage[mem=1] } } ||
{  2*{ select[type==any] order[ut] same[model] rusage[mem=1] } }@10" myjob
```

An alternative resource requirement can be used to indicate how many tasks the job requires. For example, a job may request 4 tasks on Solaris host types, or 8 tasks on Linux86 hosts types. If the `-n` option is provided at the job level, then the values specified must be consistent with the values implied by the resource requirement string:

```
bsub -R " {8*{type==LINUX86}} || {4*{type==SOLARIS}}" a.out
```

If they conflict, the job submission is rejected:

```
bsub -n 3 -R " {8*{type==LINUX86}} || {4*{type==SOLARIS}}" a.out
```

To specify alternative GPU resource requirements, use the following keywords and options:

- In the `rusage[]` section, use the `ngpus_physical` resource to request the number of physical GPUs, together with the `gmodel` option specify the GPU model, the `gmem` option to specify the amount of reserved GPU memory, and the `nvlink` option to request GPUs with NVLink connections.
- In the `span[]` section, use the `gtile` keyword to specify the number of GPUs requested on each socket.

The following example requests 4 hosts with 1 K80 GPU on each host, or 1 host with 2 P100 GPUs and 1 GPU per socket:

```
bsub -R "{4*{span[ptile=1] rusage[ngpus_physical=1:gmodel=K80]}} || {1*{span[gtile=1]
rusage[ngpus_physical=2:gmodel=P100]}}" ./app
```

-r

Reruns a job if the execution host or the system fails; it does not rerun a job if the job itself fails.

Categories

properties

Synopsis

`bsub -r`

Conflicting options

Do not use with the `-rn` option.

Description

If the execution host becomes unavailable while a job is running, specifies that the job be rerun on another host. LSF requeues the job in the same job queue with the same job ID. When an available execution host is found, reruns the job as if it were submitted new, even if the job has been checkpointed. You receive a mail message informing you of the host failure and requeuing of the job.

If the system goes down while a job is running, specifies that the job is requeued when the system restarts.

Members of a chunk job can be rerunnable. If the execution host becomes unavailable, rerunnable chunk job members are removed from the queue and dispatched to a different execution host.

Interactive jobs (**bsub -I**) are not rerunnable.

-rn

Specifies that the job is never rerunnable.

Categories

properties

Synopsis

`bsub -rn`

Conflicting options

Do not use with the `-r` option.

Description

Disables job rerun if the job was submitted to a rerunnable queue or application profile with job rerun configured. The command level job rerunnable setting overrides the application profile and queue level setting. **bsub -rn** is different from **bmod -rn**, which cannot override the application profile and queue level rerunnable job setting.

-rnc

Specifies the full path of an executable to be invoked on the first execution host when the job allocation has been modified (both shrink and grow).

Categories

properties

Synopsis

`bsub -rnc` *resize_notification_cmd*

Description

The `-rnc` option overrides the notification command specified in the application profile (if specified). The maximum length of the notification command is 4 KB.

-S

Sets a per-process (soft) stack segment size limit for each of the processes that belong to the job.

Categories

limit

Synopsis

`-S` *stack_limit*

Description

For more information, see **getrlimit**(2).

By default, the limit is specified in KB. Use **LSF_UNIT_FOR_LIMITS** in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

-s

Sends the specified signal when a queue-level run window closes.

Categories

properties

Synopsis

`bsub -s` *signal*

Description

By default, when the window closes, LSF suspends jobs running in the queue (job state becomes SSUSP) and stops dispatching jobs from the queue.

Use `-s` to specify a signal number; when the run window closes, the job is signalled by this signal instead of being suspended.

-sla

Specifies the service class where the job is to run.

Categories

properties

Synopsis

`bsub -sla` *service_class_name*

Description

If the SLA does not exist or the user is not a member of the service class, the job is rejected.

If EGO-enabled SLA scheduling is configured with **ENABLE_DEFAULT_EGO_SLA** in `lsb.params`, jobs submitted without `-sla` are attached to the configured default SLA.

You can use `-g` with `-sla`. All jobs in a job group attached to a service class are scheduled as SLA jobs. It is not possible to have some jobs in a job group not part of the service class. Multiple job groups can be created under the same SLA. You can submit additional jobs to the job group without specifying the service class name again. You cannot use job groups with resource-based SLAs that have guarantee goals.

Note: When using the `-g` with `-sla`, the job group service class overrides the service class specified with the `-sla` option. For example, if you run **bsub -g /g1 -sla sla1 myjob** to specify the `/g1` job group with the `sla1` service class,

- If there is no service class attached to the `/g1` job group, the specified `sla1` service class is ignored.
- If there is a different SLA attached to the `/g1` job group, the `/g1` job group's service class replaces the specified `sla1` service class.

LSF logs a warning message in the **mbatchd** log to notify you of these changes.

Tip: Submit your velocity, deadline, and throughput SLA jobs with a runtime limit (`-W` option) or specify **RUNLIMIT** in the queue definition in `lsb.queues` or **RUNLIMIT** in the application profile definition in `lsb.applications`. If you do not specify a runtime limit for velocity SLAs, LSF automatically adjusts the optimum number of running jobs according to the observed run time of finished jobs.

Use **bsla** to display the properties of service classes configured in `LSB_CONFDIR/`*`cluster_name`*`/configdir/lsb.serviceclasses` (see `lsb.serviceclasses`) and dynamic information about the state of each service class.

Examples

```
bsub -W 15 -sla Duncan sleep 100
```

Submit the UNIX command **sleep** together with its argument `100` as a job to the service class named `Duncan`.

The example submits and IBM PE job and assumes two hosts in cluster, `hostA` and `hostB`, each with 4 cores and 2 networks. Each network has one IB adapter with 64 windows.

-smt

Specifies the SMT mode for CSM jobs.

Categories

properties

Synopsis

`bsub -smt` *smt_value*

Description

This option is for LSF jobs that are submitted with the IBM Cluster Systems Manager (CSM) integration package.

This option has no effect if the **smt** value is specified in the **CSM_REQ** parameter of the **lsb.queues** file because that value overrides this option.

Valid values

Specify a positive integer. In addition, this must satisfy the **CSM_VALID_SMT** parameter values that are specified in the `lsb.params` file.

Default

If the **smt** value is specified in the **CSM_REQ** parameter of the **lsb.queues** file, that value overrides this `-smt` option.

If the **smt** value is not specified in the **CSM_REQ** parameter of the **lsb.queues** file, and the `-smt` option is not specified here, the default value is the first value of the **CSM_VALID_SMT** parameter in the `lsb.params` file.

-sp

Specifies user-assigned job priority that orders jobs in a queue.

Categories

properties

Synopsis

`bsub -sp` *priority*

Description

Valid values for priority are any integers between 1 and the value of the **MAX_USER_PRIORITY** parameter that is configured in the `lsb.params` file. Job priorities that are not valid are rejected. LSF administrators and queue administrators can specify priorities beyond **MAX_USER_PRIORITY** for any jobs in the queue.

Job owners can change the priority of their own jobs relative to all other jobs in the queue. LSF administrators and queue administrators can change the priority of all jobs in a queue.

Job order is the first consideration to determine job eligibility for dispatch. Jobs are still subject to all scheduling policies regardless of job priority. Jobs are scheduled based first on their queue priority first, then job priority, and lastly in first-come first-served order.

User-assigned job priority can be configured with automatic job priority escalation to automatically increase the priority of jobs that are pending for a specified period (**JOB_PRIORITY_OVER_TIME** in `lsb.params`).

When absolute priority scheduling is configured in the submission queue (the **APS_PRIORITY** parameter in the `lsb.queues` file), the user-assigned job priority is used for the JPRIORITY factor in the APS calculation.

-stage

Specifies the options for direct data staging (for example, IBM CAST burst buffer).

Categories

properties

Synopsis

```
bsub -stage "[storage=min_size[,max_size]][:in=path_to_stage_in_script]
[:out=path_to_stage_out_script]"
```

Description

The `-stage` option uses the following keywords:

storage=*min_size*[,*max_size*]
Specifies the minimum required and (optionally) maximum available storage space on the allocation.

in=*file_path*
Specifies the file path to the user stage in script. This is launched by the stage in script as specified by the **LSF_STAGE_IN_EXEC** parameter in the `lsf.conf` file.

out=*file_path*
Specifies the file path to the user stage out script. This is launched by the stage out script as specified by the **LSF_STAGE_OUT_EXEC** parameter in the `lsf.conf` file.

You can specify one or more keywords. Use a colon (`:`) to separate multiple keywords.

Example

```
bsub -stage "storage=5:in=/u/usr1/mystagein.pl:out=/home/mystagein.pl" -q bbq
myjob
```

-step_cgroup

Enables the job to create a cgroup for each job step.

Categories

properties

Synopsis

```
bsub -step_cgroup y|n
```

Description

This option is for LSF jobs that are submitted with the IBM Cluster Systems Manager (CSM) integration package.

Default

n. The job does not create a cgroup for each job step.

-T

Sets the limit of the number of concurrent threads to the specified value for the whole job.

Categories

limit

Synopsis

```
bsub -T thread_limit
```

Description

The default is no limit.

Exceeding the limit causes the job to terminate. The system sends the following signals in sequence to all processes belongs to the job: SIGINT, SIGTERM, and SIGKILL.

Examples

`bsub -T 4 myjob`

Submits `myjob` with a maximum number of concurrent threads of 4.

-t

Specifies the job termination deadline.

Categories

schedule

Synopsis

`bsub -t` [[[year:]*month*:]*day*:]*hour*:*minute*

Description

If a UNIX job is still running at the termination time, the job is sent a SIGUSR2 signal, and is killed if it does not terminate within ten minutes.

If a Windows job is still running at the termination time, it is killed immediately. (For a detailed description of how these jobs are killed, see **bkill**.)

In the queue definition, a TERMINATE action can be configured to override the **bkill** default action (see the **JOB_CONTROLS** parameter in `lsb.queues`(5)).

In an application profile definition, a **TERMINATE_CONTROL** action can be configured to override the **bkill** default action (see the **TERMINATE_CONTROL** parameter in `lsb.applications`(5)).

The format for the termination time is [[*year*:][*month*:]*day*:]*hour*:*minute* where the number ranges are as follows: year after 1970, month 1-12, day 1-31, hour 0-23, minute 0-59.

At least two fields must be specified. These fields are assumed to be *hour*:*minute*. If three fields are given, they are assumed to be *day*:*hour*:*minute*, four fields are assumed to be *month*:*day*:*hour*:*minute* and five fields are assumed to be *year*: *month*:*day*:*hour*:*minute*.

If the year field is specified and the specified time is in the past, the job submission request is rejected.

-ti

Enables automatic orphan job termination at the job level for a job with a dependency expression (set using `-w`).

Categories

schedule

Synopsis

`bsub -w '`*dependency_expression*`'[-ti]`

Conflicting options

Use only with the `-w` option.

Description

`-ti` is a sub-option of `-w`. With this sub-option, the cluster-level orphan job termination grace period is ignored (if configured) and the job can be terminated as soon as it is found to be an orphan. This option is independent of the cluster-level configuration. Therefore, if the LSF administrator did not enable **ORPHAN_JOB_TERM_GRACE_PERIOD** at the cluster level, you can still use automatic orphan job termination on a per-job basis.

-tty

When submitting an interactive job, displays output/error messages on the screen (except pre-execution output/error messages).

Categories

io

Synopsis

`bsub -I | -Ip | -IS | -ISp | -ISs | -Is | -IX [-tty]`

Conflicting options

Use only with the following options: `-I`, `-Ip`, `-IS`, `-ISp`, `-ISs`, `-Is`, `-IX`.

Description

`-tty` is a sub-option of the interactive job submission options (including `-I`, `-Ip`, `-IS`, `-ISp`, `-ISs`, `-Is`, and `-IX`).

-U

If an advance reservation has been created with the **brsvadd** command, the job makes use of the reservation.

Categories

resource, schedule

Synopsis

`bsub -U` *reservation_ID*

Description

For example, if the following command was used to create the reservation user1#0:

```
brsvadd -n 1024 -m hostA -u user1 -b 13:0 -e 18:0
Reservation "user1#0" is created
```

The following command uses the reservation:

```
bsub -U user1#0 myjob
```

The job can only use hosts reserved by the reservation `user1#0`. LSF only selects hosts in the reservation. You can use the `-m` option to specify particular hosts within the list of hosts reserved by the reservation, but you cannot specify other hosts not included in the original reservation.

If you do not specify hosts (**bsub -m**) or resource requirements (**bsub -R**), the default resource requirement is to select hosts that are of any host type (LSF assumes "type==any" instead of "type==local" as the default select string).

If you later delete the advance reservation while it is still active, any pending jobs still keep the "type==any" attribute.

A job can only use one reservation. There is no restriction on the number of jobs that can be submitted to a reservation; however, the number of slots available on the hosts in the reservation may run out. For example, reservation `user2#0` reserves 128 slots on `hostA`. When all 128 slots on `hostA` are used by jobs referencing `user2#0`, `hostA` is no longer available to other jobs using reservation `user2#0`. Any single user or user group can have a maximum of 100 reservation IDs.

Jobs referencing the reservation are killed when the reservation expires. LSF administrators can prevent running jobs from being killed when the reservation expires by changing the termination time of the job using the reservation (**bmod -t**) before the reservation window closes.

To use an advance reservation on a remote host, submit the job and specify the remote advance reservation ID. For example:

```
bsub -U user1#01@cluster1
```

In this example, it is assumed that the default queue is configured to forward jobs to the remote cluster.

-u

Sends mail to the specified email destination.

Categories

notify

Synopsis

`bsub -u` *mail_user*

`bsub -u "`*user@example.com* ... `"`

Description

The email destination can be an email address or an OS user account. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) in a Windows command line or a double backslash (*DOMAIN_NAME**user_name*) in a UNIX command line.

To specify multiple email addresses, use quotation marks and separate each address with a space. The total length of the address string cannot be longer than 511 characters.

-ul

Passes the current operating system user shell limits for the job submission user to the execution host.

Categories

limit

Synopsis

`bsub -ul`

Description

User limits cannot override queue hard limits. If user limits exceed queue hard limits, the job is rejected.

Restriction: UNIX and Linux only. `-ul` is not supported on Windows.

The following **bsub** options for job-level runtime limits override the value of the user shell limits:

- Per-process (soft) core file size limit (`-C`)
- CPU limit (`-c`)
- Per-process (soft) data segment size limit (`-D`)
- File limit (`-F`)
- Per-process (soft) memory limit (`-M`)
- Process limit (`-p`)
- Per-process (soft) stack segment size limit (`-S`)
- Limit of the number of concurrent threads (`-T`)

- Total process virtual memory (swap space) limit (`-v`)
- Runtime limit (`-W`)

LSF collects the user limit settings from the user's running environment that are supported by the operating system, and sets the value to submission options if the value is not unlimited. If the operating system has other kinds of shell limits, LSF does not collect them. LSF collects the following operating system user limits:

- CPU time in milliseconds
- Maximum file size
- Data size
- Stack size
- Core file size
- Resident set size
- Open files
- Virtual (swap) memory
- Process limit
- Thread limit

-v

Sets the total process virtual memory limit to the specified value for the whole job.

Categories

limit

Synopsis

`bsub -v` *swap_limit*

Description

The default is no limit. Exceeding the limit causes the job to terminate.

By default, the limit is specified in KB. Use **LSF_UNIT_FOR_LIMITS** in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

-W

Sets the runtime limit of the job.

Categories

limit

Synopsis

`bsub -W` [*hour*`:`]*minute*[*/host_name* | */host_model*]

Description

If a UNIX job runs longer than the specified run limit, the job is sent a SIGUSR2 signal, and is killed if it does not terminate within ten minutes. If a Windows job runs longer than the specified run limit, it is killed immediately. (For a detailed description of how these jobs are killed, see **bkill**.)

In the queue definition, a **TERMINATE** action can be configured to override the **bkill** default action (see the **JOB_CONTROLS** parameter in `lsb.queues`).

In an application profile definition, a **TERMINATE_CONTROL** action can be configured to override the **bkill** default action (see the **TERMINATE_CONTROL** parameter in `lsb.applications`).

If you want to provide LSF with an estimated run time without killing jobs that exceed this value, submit the job with `-We`, or define the **RUNTIME** parameter in `lsb.applications` and submit the job to that application profile. LSF uses the estimated runtime value for scheduling purposes only.

The run limit is in the form of [*hour*`:`]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The run limit you specify is the normalized run time. This is done so that the job does approximately the same amount of processing, even if it is sent to host with a faster or slower CPU. Whenever a normalized run time is given, the actual time on the execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

If **ABS_RUNLIMIT=Y** is defined in `lsb.params`, the runtime limit and the runtime estimate are not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted with a runtime limit or runtime estimate.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert '`/`' between the run limit and the host name or model name.

If no host or host model is given, LSF uses the default runtime normalization host defined at the queue level (**DEFAULT_HOST_SPEC** in `lsb.queues`) if it has been configured; otherwise, LSF uses the default CPU time normalization host defined at the cluster level (**DEFAULT_HOST_SPEC** in `lsb.params`) if it has been configured; otherwise, LSF uses the submission host.

For LSF multicluster capability jobs, if no other CPU time normalization host is defined and information about the submission host is not available, LSF uses the host with the largest CPU factor (the fastest host in the cluster).

If the job also has termination time specified through the **bsub -t** option, LSF determines whether the job can actually run for the specified length of time allowed by the run limit before the termination time. If not, then the job is aborted.

If the **IGNORE_DEADLINE** parameter is set in `lsb.queues`, this behavior is overridden and the run limit is ignored.

Jobs submitted to a chunk job queue are not chunked if the run limit is greater than 30 minutes.

Examples

```
bsub -W 15 -sla Duncan sleep 100
```

Submit the UNIX command **sleep** together with its argument `100` as a job to the service class named `Duncan`.

-We

Specifies an estimated run time for the job.

Categories

limit

Synopsis

`bsub -We` [*hour*`:`]*minute*[*/host_name* | */host_model*]

Description

LSF uses the estimated value for job scheduling purposes only, and does not kill jobs that exceed this value unless the jobs also exceed a defined runtime limit. The format of runtime estimate is same as run limit set by the `-W` option.

Use **JOB_RUNLIMIT_RATIO** in `lsb.params` to limit the runtime estimate users can set. If **JOB_RUNLIMIT_RATIO** is set to 0 no restriction is applied to the runtime estimate.

The job-level runtime estimate setting overrides the **RUNTIME** setting in an application profile in `lsb.applications`.

-w

LSF does not place your job unless the dependency expression evaluates to TRUE.

Categories

schedule

Synopsis

`bsub -w '`*dependency_expression*`'[-ti]`

Description

-w '*dependency_expression*' [-ti]

If you specify a dependency on a job that LSF cannot find (such as a job that has not yet been submitted), your job submission fails.

The dependency expression is a logical expression composed of one or more dependency conditions. To make dependency expression of multiple conditions, use the following logical operators:

&& (AND)

|| (OR)

! (NOT)

Use parentheses to indicate the order of operations, if necessary.

Enclose the dependency expression in single quotes (') to prevent the shell from interpreting special characters (space, any logic operator, or parentheses). If you use single quotes for the dependency expression, use double quotes (") for quoted items within it, such as job names.

In a Windows environment with multiple job dependencies, use only double quotes.

In dependency conditions, job names specify only your own jobs. By default, if you use the job name to specify a dependency condition, and more than one of your jobs has the same name, all of your jobs that have that name must satisfy the test. If **JOB_DEP_LAST_SUB** in `lsb.params` is set to 1, the test is done on the job submitted most recently.

Use double quotes (") around job names that begin with a number. In the job name, specify the wildcard character asterisk (*) at the end of a string, to indicate all jobs whose name begins with the string. For example, if you use `jobA*` as the job name, it specifies jobs named `jobA`, `jobA1`, `jobA_test`, `jobA.log`, etc.

Use the * with dependency conditions to define one-to-one dependency among job array elements such that each element of one array depends on the corresponding element of another array. The job array size must be identical.

For example:

```
bsub -w "done(myarrayA[*])" -J "myArrayB[1-10]" myJob2
```

indicates that before element 1 of `myArrayB` can start, element 1 of `myArrayA` must be completed, and so on.

You can also use the * to establish one-to-one array element dependencies with **bmod** after an array has been submitted.

If you want to specify array dependency by array name, set **JOB_DEP_LAST_SUB** in `lsb.params`. If you do not have this parameter set, the job is rejected if one of your previous arrays has the same name but a different index.

In dependency conditions, the variable *op* represents one of the following relational operators:

>

>=

<

<=

==

!=

Use the following conditions to form the dependency expression. Where *job_name* is mentioned, LSF refers to the oldest job of *job_name* in memory.

done(*job_ID* |"*job_name*" ...)

The job state is DONE.

ended(*job_ID* | "*job_name*")

The job state is EXIT or DONE.

exit(*job_ID* | "*job_name*" [,[*operator*] *exit_code*])

The job state is EXIT, and the job's exit code satisfies the comparison test.

If you specify an exit code with no operator, the test is for equality (== is assumed).

If you specify only the job, any exit code satisfies the test.

external(*job_ID* | "*job_name*", "*status_text*")

The job has the specified job status. (Commands **bstatus** and **bpost** set, change, and retrieve external job status messages.)

If you specify the first word of the job status description (no spaces), the text of the job's status begins with the specified word. Only the first word is evaluated.

***job_ID* | "*job_name*"**

If you specify a job without a dependency condition, the test is for the DONE state (LSF assumes the “done” dependency condition by default).

numdone(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the DONE state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numended(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the DONE or EXIT states satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numexit(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the EXIT state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numhold(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the PSUSP state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numpend(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the PEND state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numrun(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the RUN state satisfies the test. Use * (with no operator) to specify all the jobs in the array.

numstart(*job_ID*, *operator number* | *)

For a job array, the number of jobs in the RUN, USUSP, or SSUSP states satisfies the test. Use * (with no operator) to specify all the jobs in the array.

post_done(*job_ID* | "*job_name*")

The job state is POST_DONE (post-execution processing of the specified job has completed without errors).

post_err(*job_ID* | "*job_name*")

The job state is POST_ERR (post-execution processing of the specified job has completed with errors).

started(*job_ID* | "*job_name*")

The job state is:

- USUSP, SSUSP, DONE, or EXIT
- RUN and the job has a pre-execution command (**bsub -E**) that is done.

-wa

Specifies the job action to be taken before a job control action occurs.

Categories

properties

Synopsis

```
bsub -wa 'signal'
```

Description

A job warning action must be specified with a job action warning time in order for job warning to take effect.

If -wa is specified, LSF sends the warning action to the job before the actual control action is taken. This allows the job time to save its result before being terminated by the job control action.

The warning action specified by -wa option overrides **JOB_WARNING_ACTION** in the queue. **JOB_WARNING_ACTION** is used as the default when no command line option is specified.

Examples

The following specifies that 2 minutes before the job reaches its runtime limit, a URG signal is sent to the job:

```
bsub -W 60 -wt '2' -wa 'URG' myjob
```

-wt

Specifies the amount of time before a job control action occurs that a job warning action is to be taken.

Categories

properties

Synopsis

`bsub -wt '[`*hour*`:]`*minute*`'`

Description

Job action warning time is not normalized.

A job action warning time must be specified with a job warning action in order for job warning to take effect.

The warning time specified by the **bsub -wt** option overrides **JOB_ACTION_WARNING_TIME** in the queue. **JOB_ACTION_WARNING_TIME** is used as the default when no command line option is specified.

Examples

The following specifies that 2 minutes before the job reaches its runtime limit, an URG signal is sent to the job:

```
bsub -W 60 -wt '2' -wa 'URG' myjob
```

-XF

Submits a job using SSH X11 forwarding.

Categories

properties

Synopsis

```
bsub -XF
```

Conflicting options

Do not use with the following options: `-IX`, `-K`, `-r`.

Description

A job submitted with SSH X11 forwarding cannot be used with job arrays, job chunks, or user account mapping.

Jobs with SSH X11 forwarding cannot be checked or modified by an **esub**.

Use **-XF** with **-I** to submit an interactive job using SSH X11 forwarding. The session is displayed throughout the job lifecycle.

Note: LSF does not support SSH X11 forwarding with the LSF multicluster capability.

bjobs -l displays job information, including any jobs submitted with SSH X11 forwarding.

bhist -l displays historical job information, including any jobs submitted with SSH X11 forwarding.

Optionally, specify **LSB_SSH_XFORWARD_CMD** in `lsf.conf`. You can replace the default value with an SSH command (full PATH and options allowed).

For more information about **LSB_SSH_XFORWARD_CMD**, see the *LSF Configuration Reference*.

Troubleshooting

Enable the following parameters in `lsf.conf`::

- **LSF_NIOS_DEBUG=1**
- **LSF_LOG_MASK="LOG_DEBUG"**

SSH X11 forwarding must be already working outside LSF.

-x

Puts the host running your job into exclusive execution mode.

Categories

schedule

Synopsis

```
bsub -x
```

Description

In exclusive execution mode, your job runs by itself on a host. It is dispatched only to a host with no other jobs running, and LSF does not send any other jobs to the host until the job completes.

To submit a job in exclusive execution mode, the queue must be configured to allow exclusive jobs.

When the job is dispatched, **bhosts** reports the host status as closed_Excl, and **lsload** reports the host status as lockU.

Until your job is complete, the host is not selected by LIM in response to placement requests made by **lsplace**, **lsrun** or **lsgrun** or any other load sharing applications.

You can force other jobs to run on the host by using the `-m` *host_name* option of **brun(1)** to explicitly specify the locked host.

You can force LIM to run other interactive jobs on the host by using the `-m` *host_name* option of **lsrun** or **lsgrun** to explicitly specify the locked host.

-Zs

Spools a job command file to the directory specified by the **JOB_SPOOL_DIR** parameter in `lsb.params`, and uses the spooled file as the command file for the job.

Categories

properties, script

Synopsis

```
bsub -Zs
```

Description

By default, the command file is spooled to LSB_SHAREDIR/*cluster_name*/`lsf_cmddir`. If the `lsf_cmddir` directory does not exist, LSF creates it before spooling the file. LSF removes the spooled file when the job completes.

If **JOB_SPOOL_DIR** is specified, the `-Zs` option spools the command file to the specified directory and uses the spooled file as the input file for the job.

JOB_SPOOL_DIR can be any valid path up to a maximum length up to 4094 characters on UNIX and Linux or up to 255 characters for Windows.

JOB_SPOOL_DIR must be readable and writable by the job submission user, and it must be shared by the master host and the submission host. If the specified directory is not accessible or does not exist, **bsub -Zs** cannot write to the default directory LSB_SHAREDIR/*cluster_name*/lsf_cmddir and the job fails.

The -Zs option is not supported for embedded job commands because LSF is unable to determine the first command to be spooled in an embedded job command.

command

Specifies the command and arguments used for the job submission.

Synopsis

bsub [*options*] *command* [*arguments*]

Description

The job can be specified by a command line argument command, or through the standard input if the command is not present on the command line. The *command* can be anything that is provided to a UNIX Bourne shell (see **sh**(1)). command is assumed to begin with the first word that is not part of a **bsub** option. All arguments that follow *command* are provided as the arguments to the *command*. Use single quotation marks around the expression if the command or arguments contain special characters.

The job command can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. If no job name is specified with **-J**, **bjobs**, **bhist** and **bacct** displays the command as the job name.

If the job is not given on the command line, **bsub** reads the job commands from standard input. If the standard input is a controlling terminal, the user is prompted with bsub> for the commands of the job. The input is terminated by entering CTRL-D on a new line. You can submit multiple commands through standard input.

The commands are executed in the order in which they are given. **bsub** options can also be specified in the standard input if the line begins with #BSUB; for example, #BSUB -x. If an option is given on both the **bsub** command line, and in the standard input, the command line option overrides the option in the standard input. The user can specify the shell to run the commands by specifying the shell path name in the first line of the standard input, such as #!/bin/csh. If the shell is not given in the first line, the Bourne shell is used. The standard input facility can be used to spool a user's job script; such as bsub < script.

Examples

```
bsub sleep 100
```

Submit the UNIX command **sleep** together with its argument 100 as a job.

```
bsub my_script
```

Submit my_script as a job. Since my_script is specified as a command line argument, the my_script file is not spooled. Later changes to the my_script file before the job completes may affect this job.

```
bsub < default_shell_script
```

where default_shell_script contains:

```
sim1.exe
sim2.exe
```

The file default_shell_script is spooled, and the commands are run under the Bourne shell since a shell specification is not given in the first line of the script.

```
bsub < csh_script
```

where `csh_script` contains:

```
#!/bin/csh
sim1.exe
sim2.exe
```

`csh_script` is spooled and the commands are run under **/bin/csh**.

`bsub -q night < my_script`

where `my_script` contains:

```
#!/bin/sh
#BSUB -q test
#BSUB -m "host1 host2" # my default candidate hosts
#BSUB -f "input > tmp" -f "output << tmp"
#BSUB -D 200 -c 10/host1
#BSUB -t 13:00
#BSUB -k "dir 5"
sim1.exe
sim2.exe
```

The job is submitted to the `night` queue instead of `test`, because the command line overrides the script.

```
bsub> sleep 1800
bsub> my_program
bsub> CTRL-D
```

The job commands are entered interactively.

job_script

Specifies the job script for the **bsub** command to load, parse, and run from the command line.

Synopsis

bsub [*options*] *job_script*

Description

The **LSB_BSUB_PARSE_SCRIPT** parameter must be set to Y in the `lsf.conf` file to specify a job script.

The job script must be an ASCII text file and not a binary file. The job script does not have to be an executable file because the contents of the job script are parsed and run as part of the job.

Use the #BSUB imperative (in upper case letters) at the beginning of each line to specify job submission options in the script.

Example

The following script (`myscript.sh`) uses the #BSUB imperative to run the **myjob1 arg1** and **myjob2 arg2** commands with the **bsub -n 2** and **-P myproj** options:

```
#!/bin/sh
#BSUB -n 2
myjob1 arg1
myjob2 arg2
#BSUB -P myproj
```

Run the following command to use this job script:

```
bsub myscript.sh
```

LSB_DOCKER_PLACE_HOLDER

Submits a Docker job with a Docker entry point image, but no command.

Synopsis

`bsub` [*options*] `LSB_DOCKER_PLACE_HOLDER`

Description

As for all Docker container jobs, you must use the `-app` option to specify an application profile that is configured to run Docker jobs. The Docker execution driver must be defined in this application profile in the `lsb.applications` file.

Example

```
bsub -app docker1 LSB_DOCKER_PLACE_HOLDER
```

-h

Displays a description of the specified category, command option, or sub-option to `stderr` and exits.

Synopsis

`bsub -h[elp]` [*category* ...] [*option* ...]

Description

You can abbreviate the `-help` option to `-h`.

Run **bsub -h** (or **bsub -help**) without a command option or category name to display the **bsub** command description.

Examples

`bsub -h io`

Displays a description of the `io` category and the **bsub** command options belonging to this category.

`bsub -h -app`

Displays a detailed description of the **bsub -app** option.

-V

Prints LSF release version to `stderr` and exits.

Synopsis

`bsub -V`

Conflicting options

Do not use with any other option except `-h` (**bsub -h -V**).

Description

Submits a job for execution and assigns it a unique numerical job ID.

Runs the job on a host that satisfies all requirements of the job, when all conditions on the job, host, queue, application profile, and cluster are satisfied. If LSF cannot run all jobs immediately, LSF scheduling policies determine the order of dispatch. Jobs are started and suspended according to the current system load.

Sets the user's execution environment for the job, including the current working directory, file creation mask, and all environment variables, and sets LSF environment variables before the job starts.

When a job is run, the command line and `stdout/stderr` buffers are stored in the directory home_directory/`.lsbatch` on the execution host. If this directory is not accessible, `/tmp/.lsbtmp` user_ID is used as the job's home directory. If the current working directory is under the home directory on the submission host, then the current working directory is also set to be the same relative directory under the home directory on the execution host.

By default, if the current working directory is not accessible on the execution host, LSF finds a working direction to run the job in the following order:

1. *$PWD*
2. Strip `/tmp_mnt` if it is exists in the path
3. Replace the first component with a key in `/etc/auto.master` and try each key
4. Replace the first 2 components with a key in `/etc/auto.master` and try for each key
5. Strip the first level of the path and try the rest (for example, if the current working directory is `/abc/x/y/z`, try to change directory to the path `/x/y/z`)
6. `/tmp`

If the environment variable **LSB_EXIT_IF_CWD_NOTEXIST** is set to Y and the current working directory is not accessible on the execution host, the job exits with the exit code 2.

If no command is supplied, **bsub** prompts for the command from the standard input. On UNIX, the input is terminated by entering CTRL-D on a new line. On Windows, the input is terminated by entering CTRL-Z on a new line.

To kill a job that is submitted with **bsub**, use **bkill**.

Use **bmod** to modify jobs that are submitted with **bsub**. **bmod** takes similar options to **bsub**.

Jobs that are submitted to a chunk job queue with the following options are not chunked; they are dispatched individually:

- **-I** (interactive jobs)
- **-c** (jobs with CPU limit greater than 30)
- **-W** (jobs with run limit greater than 30 minutes)

To submit jobs from UNIX to display GUIs through Microsoft Terminal Services on Windows, submit the job with **bsub** and define the environment variables **LSF_LOGON_DESKTOP=1** and **LSB_TSJOB=1** on the UNIX host. Use **tssub** to submit a Terminal Services job from Windows hosts. See *Using LSF on Windows* for more details.

If the parameter **LSB_STDOUT_DIRECT** in `lsf.conf` is set to Y or y, and you use the `-o` or `-oo` option, the standard output of a job is written to the file you specify as the job runs. If **LSB_STDOUT_DIRECT** is not set, and you use `-o` or `-oo`, the standard output of a job is written to a temporary file and copied to the specified file after the job finishes. **LSB_STDOUT_DIRECT** is not supported on Windows.

Default behavior

LSF assumes that uniform user names and user ID spaces exist among all the hosts in the cluster. That is jobs run under on the execution host the same users account as the job was submitted with. For situations where nonuniform user names and user ID spaces exist, account mapping must be used to determine the account that is used to run a job.

bsub uses the command name as the job name. Quotation marks are significant.

Options that are related to file names and job spooling directories support paths that contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

Options that are related to command names and job names can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

Options for the following resource usage limits are specified in KB:

- Core limit (`-C`)
- Memory limit (`-M`)
- Stack limit (`-S`)
- Swap limit (`-v`)

Use **LSF_UNIT_FOR_LIMITS** in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

If fairshare is defined and you belong to multiple user groups, the job is scheduled under the user group that allows the quickest dispatch.

The job is not checkpointable.

bsub automatically selects an appropriate queue. If you defined a default queue list by setting **LSB_DEFAULTQUEUE** environment variable, the queue is selected from your list. If **LSB_DEFAULTQUEUE** is not defined, the queue is selected from the system default queue list that is specified by the LSF administrator with the **DEFAULT_QUEUE** parameter in `lsb.params`.

LSF tries to obtain resource requirement information for the job from the remote task list that is maintained by the load sharing library. If the job is not listed in the remote task list, the default resource requirement is to run the job on a host or hosts that are of the same host type as the submission host.

bsub assumes that only one processor is requested.

bsub does not start a login shell but runs the job file under the execution environment from which the job was submitted.

The input file for the job is `/dev/null` (no input).

bsub sends mail to you when the job is done. The default destination is defined by **LSB_MAILTO** in `lsf.conf`. The mail message includes the job report, the job output (if any), and the error message (if any).

bsub charges the job to the default project. The default project is the project that you define by setting the environment variable **LSB_DEFAULTPROJECT**. If you do not set **LSB_DEFAULTPROJECT**, the default project is the project that is specified by the LSF administrator with **DEFAULT_PROJECT** parameter in `lsb.params`. If **DEFAULT_PROJECT** is not defined, then LSF uses default as the default project name.

Output

If the job is successfully submitted, **bsub** displays the job ID and the queue that the job was submitted to.

Limitations

When account mapping is used, the command **bpeek** does not work. File transfer with the `-f` option to **bsub** requires **rcp** to be working between the submission and execution hosts. Use the `-N` option to request mail, or the `-o` and `-e` options to specify an output file and error file.

Submitting jobs with SSH

Secure Shell (SSH) is a network protocol that provides confidentiality and integrity of data using a secure channel between two networked devices.

SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.

SSH is typically used to log into a remote machine and execute commands, but it also supports tunneling, forwarding arbitrary TCP ports and X11 connections. SSH uses a client-server protocol.

SSH uses private/public key pairs to log into another host. Users no longer have to supply a password every time they log on to a remote host.

SSH is used when running any of the following jobs:

- Remote log on to a lightly loaded host (**lslogin**)
- An interactive job (**bsub -IS** | **-ISp** | **ISs**)
- An interactive X-window job with X11 forwarding (**bsub -XF**)
- An interactive X-window job, without X11 forwarding (**bsub -IX**)
- An externally submitted job (**esub**)

SSH only supports UNIX for submission and execution hosts. The display host can be any operating system.

Depending on your requirements for X-Window jobs, you can choose either **bsub -XF** (recommended) or **bsub -IX**. Both options encrypt the X-Server and X-Clients.

SSH X11 forwarding (**bsub -XF**) has the following benefits:

- Any password required can be typed in when needed.
- Does not require the X-Server host to have the SSH daemon installed.

SSH X11 forwarding has the following drawbacks:

- The user must enable X11 forwarding in the client.
- Submission and execution hosts must be UNIX hosts.

SSH X11 forwarding has the following dependencies:

- OpenSSH 3.9p1 and up is supported.

  OpenSSL 0.9.7a and up is supported.
- You must have SSH correctly installed on all hosts in the cluster.
- You must use an SSH client to log on to the submission host from the display host.
- You must install and run the X-Server program on the display host.

SSH X11 forwarding has the following limitations:

- You cannot run with **bsub -K**, **-IX**, or **-r**.
- You cannot **bmod** a job submitted with X11 forwarding.
- Cannot be used with job arrays, job chunks, or user account mapping.
- Jobs submitted with X11 forwarding cannot be checked or modified by **esubs**.
- Can only run on UNIX hosts (submission and execution hosts).

SSH interactive X-Window (**bsub -IX**) has the following benefits:

- The execution host contacts the X-Server host directly (no user steps required).
- Hosts can be any OS that OpenSSH supports.

SSH interactive X-Window has the following drawbacks:

- Requires the SSH daemon installed on the X-Server host.
- Must use private keys with no passwords set.

SSH interactive X-Window has the following dependencies:

- You must have OpenSSH correctly installed on all hosts in the cluster.
- You must generate public/private key pairs and add the content of the public key to the authorized_keys file on remote hosts. For more information, refer to your SSH documentation.
- For X-window jobs, you must set the DISPLAY environment variable to *X-serverHost*`:0.0`, where *X-serverHost* is the name of the X-window server. Ensure that the X-server can access itself. Run, for example, **xhost +localhost**.

SSH interactive X-Window has the following limitations:

- Cannot be used with job arrays or job chunks.

- Private user keys must have no password set.
- You cannot run with **-K** , **-r**, or **-XF**.

See also

bjobs, **bkill**, **bqueues**, **bhosts**, **bmgroup**, **bmod**, **bchkpnt**, **brestart**, **bgadd**, **bgdel**, **bjgroup**, **sh**, **getrlimit**, **sbrk**, libckpt.a, lsb.users, lsb.queues, lsb.params, lsb.hosts, lsb.serviceclasses, **mbatchd**

Chapter 61. bswitch

Switches unfinished jobs from one queue to another.

Synopsis

`bswitch` [`-a " `*application_name* [([*argument*[,*argument*...]])]..."] [`-J` *job_name*] [`-m` *host_name* | `-m` *host_group* | `-m` *compute_unit*] [`-q` *queue_name*] [`-u` *user_name* | `-u` *user_group* | `-u all`] *destination_queue* [`0`]

`bswitch` *destination_queue* [*job_ID* | "*job_ID*[*index_list*]"] ...

`bswitch` [`-h` | `-V`]

Description

Switches one or more of your unfinished jobs to the specified queue. LSF administrators and root can switch jobs that are submitted by other users.

By default, switches one job, the most recently submitted job, or the most recently submitted job that also satisfies other specified options (`-m`, `-q`, `-u`, or `-J`). Specify `0` (zero) to switch multiple jobs.

The switch operation can be done only if a specified job is acceptable to the new queue as if it were submitted to it, and, in case the job that was dispatched to a host, if the host can be used by the new queue. If the switch operation is unsuccessful, the job stays where it is. Switched jobs use the successful application exit values (that is, the exit codes specified by the **SUCCESS_EXIT_VALUES** parameter) from the new queue.

If the parameter **DEFAULT_USER_GROUP** in the `lsb.params` file is defined, a job that is switched to a queue where it cannot run (without shares in a fairshare queue, for example) is transferred to the default user group so the job can run.

If a switched job was not dispatched, then its behavior is as if it were submitted to the new queue in the first place.

If a switched job was dispatched, then it is controlled by the `loadSched` and `loadStop` vectors and other configuration parameters of the new queue, but its nice value and resource limits remain the same. Also, the switched job is controlled by the **PRIORITY** and **RUN_WINDOW** configuration parameters of the new queue.

Members of a chunk job can be switched to another queue. Running chunk job members are removed from the chunk and switched; all other jobs in `WAIT` state are requeued to `PEND` state. For chunk jobs in `WAIT` state, only the `WAIT` job is removed from the chunk and switched, and requeued to `PEND`.

The **bswitch** command is useful to change a job's attributes that are inherited from the queue.

The **bswitch** command can switch resizable jobs between queues regardless of job state. After the job is switched, the parameters in new queue apply, including threshold configuration, run limit, CPU limit, queue-level resource requirements, and so on. Multi-phase `rusage` string resource requirements can be switched in the middle of a phase.

When you switch a job between fairshare queues by using decayed run time to calculate dynamic priority, the decayed run time is switched. If the old queue did not decay the run time, the non-decayed run time is switched over; if the new queue does not decay run time, the undecayed run time is switched over.

When you switch a pending job to a queue with limits set by the parameter **RESRSV_LIMIT** in the `lsb.queues` file, the job's `rusage` values must be within the set limits or the job cannot be switched. When you switch a running job to a queue with limits set by the parameter **RESRSV_LIMIT**, the job's maximum `rusage` values cannot exceed the maximums set by the **RESRSV_LIMIT** parameter, but the job's `rusage` values can be lower than the minimum values.

By default, the job's effective resource requirements are not changed when you use the **bswitch** command. The effective resource requirement string for scheduled jobs represents the resource

requirement that is used by the scheduler to make a dispatch decision. If the **BSWITCH_MODIFY_RUSAGE** parameter is enabled and you run the **bswitch** command, the job's effective resource requirements are changed according to the new combined resource requirements.

When you switch a job that is auto-attached to a guaranteed service class, the auto-attachment is reevaluated if required.

Options

0
(Zero). Switches multiple jobs. Switches all the jobs that satisfy other specified options (`-m`, `-q`, `-u`, and `-J`).

-a "*application_name*(arg1[,arg2 ...]) ..."
Specifies an application-specific **eswitch** executable file that you want LSF to associate with the switch request.

This option functions the same as the **bsub -a** option, except that it controls `eswitch` files instead of `esub`/`epsub` files.

-J *job_name*
Switches only jobs that have the specified job name.

The job name can be up to 4094 characters long. Job names are not unique.

The wildcard character (`*`) can be used anywhere within a job name, but it cannot appear within an array index. For example, the pattern `job*` returns `jobA` and `jobarray[1]`. The `*AAA*[1]` pattern returns the first element in job arrays with names that contain AAA. However, the pattern `job1[*]` does not return anything since the wildcard is within the array index.

-m *host_name* | -m *host_group* | -m *compute_unit*
Switches only jobs that are dispatched to the specified host, host group, or compute unit.

-q *queue_name*
Switches only jobs in the specified queue.

If the job has a RUNLIMIT defined in `lsb.queues`, LSF will consider the new run limit while scheduling the AR jobs.

-u *user_name* | -u *user_group* | -u all
Switches only jobs that are submitted by the specified user, or all users if you specify the keyword all. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) on a Windows command prompt or a double backslash (*DOMAIN_NAME**user_name*) on a UNIX or Linux command line.

If you specify a user group, switches jobs submitted by all users in the group.

destination_queue
Required. Specify the queue to which the job is to be moved.

job_ID ... |"job_ID[index_list]" ...
Switches only the specified jobs.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Limitations

You cannot switch a LSF multicluster capability job.

See also

bhosts, **bjobs**, **bqueues**, **bsub**, **bugroup**

Chapter 62. btop

Moves a pending job relative to the first job in the queue.

Synopsis

`btop [-u] job_ID | "job_ID[index_list]" [position]`

`btop [-h | -V]`

Description

Changes the queue position of a pending job or a pending job array element to affect the order in which jobs are considered for dispatch.

By default, LSF dispatches jobs in a queue in the order of their arrival (that is, first come, first served), subject to availability of suitable server hosts.

Use the **btop** command to manually change the order in which jobs are considered for dispatch. Users can operate only on their own jobs. LSF administrators can operate on any user's jobs. Users can change the relative position only for their own jobs.

If the LSF administrator uses the **btop** command, the selected job is moved before the first job with the same priority that is submitted to the queue. The positions of all users' jobs in the queue can be changed by the LSF administrator.

If a regular user uses the **btop** command, the selected job is moved before the first job with the same priority that is submitted by the user to the queue. Pending jobs are displayed by the **bjobs** command in the order in which they are considered for dispatch.

You can use the **btop** command to change the dispatch order of your jobs that was scheduled through a fairshare policy. If the LSF administrator uses the **btop** command to move a job that was scheduled through a fairshare policy, the job is not subject to further fairshare scheduling unless the same job is later moved by the LSF administrator with the **bbot** command. In this case, the job is scheduled again by using the same fairshare policy.

To prevent users from changing the queue position of a pending job with the **btop** command, configure the **JOB_POSITION_CONTROL_BY_ADMIN=Y** in the `lsb.params` file.

You cannot run the **btop** command on jobs that are pending in an absolute priority scheduling (APS) queue.

Options

-u
Optional. When specified, allows jobs to be moved relative to the normal user's own job list. This option can only be used by the LSF administrator. If used by a normal user, this option is ignored.

***job_ID* | "*job_ID[index_list]*"**
Required. Job ID of the job or of the job array on which to operate.

For a job array, the index list, the square brackets, and the quotation marks are required. An index list is used to operate on a job array. The index list is a comma-separated list whose elements have the syntax *start_index*`[-`*end_index*`[:`*step*`]]` where *start_index*, *end_index*, and *step* are positive integers. If the step is omitted, a step of one is assumed. The job array index starts at one. The maximum job array index is 1000. All jobs in the array have the same job ID and parameters. Each element of the array is distinguished by its array index.

position
Optional. The *position* argument can be specified to indicate where in the queue the job is to be placed. The *position* option is a positive number that indicates the target position of the job from the beginning of the queue. The positions are relative to only the applicable jobs in the queue,

depending on whether the invoker is a regular user or the LSF administrator. The default value of 1 means the position is before all the other jobs in the queue that have the same priority.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

bbot, **bjobs**, **bswitch**

Chapter 63. bugroup

Displays information about user groups.

Synopsis

`bugroup` [`-l`] [`-r`] [`-w`] [*user_group* ...]

`bugroup` [`-h` | `-V`]

Description

Displays user groups, user names, APS user group (UG) factors, user shares, and group administrators for each group. Group administrators are expanded to show individual user names even if a user group is the configured administrator. Group administrator rights inherited from member subgroups are also shown.

The default is to display information about all user groups.

Options

-l
Displays information in a long multi-line format. Also displays share distribution if shares are configured.

-r
Expands the user groups recursively. The expanded list contains only user names; it does not contain the names of subgroups. Duplicate user names are listed only once.

-w
Wide format. Displays user and user group names without truncating fields.

user_group ...
Displays only information about the specified user groups. Do not use quotation marks when you specify multiple user groups.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Output

In the list of users, a name followed by a slash (/) indicates a subgroup.

Files

User groups, groups administrators, APS priority factors, and user shares are defined in the configuration file `lsb.users`.

See also

bmgroup, **busers**, `lsb.users`

Chapter 64. busers

Displays information about users and user groups.

Synopsis

`busers` [`-alloc`] [`-w`] [*user_name* ... | *user_group* ... | `all`]

`busers` [`-h` | `-V`]

Description

By default, displays information about the user who runs the command.

When a resizable job has a resize allocation request, the **busers** command displays pending requests. When LSF adds more resources to a running resizable job, the **busers** command decreases pending job counts and displays the added resources. When LSF removes resources from a running resizable job, the **busers** command displays the updated resources.

Options

-alloc
Shows counters for slots in RUN, SSUSP, USUSP, and RSV state. The slot allocation is different depending on whether the job is an exclusive job or not.

***user_name* ... | *user_group* ... | all**
Displays information about the specified users or user groups, or about all users if you specify the `all` option. To specify a Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) on a Windows command prompt or a double backslash (*DOMAIN_NAME**user_name*) on a UNIX ore Linux command line.

-w
Prints pending job thresholds for users and user groups and exits. Output shows PEND, MPEND, PJOBS, and MPJOBS fields.

-o

Sets the customized output format. Specify which **busers** fields to display, and in which order.

The **busers -o** option overrides the **LSB_BUSERS_FORMAT** environment variable, which overrides the **LSB_BUSERS_FORMAT** setting in `lsf.conf`.

The following are the field names used to specify the **busers** fields to display:

- user (the name of the user or user group)
- jl/p
- max
- nstart (the current number of starting tasks for all of a users' jobs)
- pend
- run
- ssusp
- ususp
- rsv
- njobs
- pjobs
- mpend
- mpjobs

- priority
- ngpus (the number of physical GPUs that the users or user groups are using)
- ngpus_shared (the number of physical GPUs that the users or user groups are using in shared mode)
- ngpus_excl (the number of physical GPUs that the users or user groups are using in exclusive mode)
- ngpus_shared_jexcl (the number of physical GPUs that the users or user groups are using in shared mode, but the jobs of the user is exclusive)

Field names are case-sensitive.

For example,

```
busers -o "user ngpus_alloc ngpus_excl_alloc ngpus_shared_alloc
ngpus_shared_jexcl_alloc"
```

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Output

A listing of the users and user groups is displayed with the following fields:

USER/GROUP
The name of the user or user group.

JL/P
The maximum number of job slots that can be processed simultaneously for the specified users on each processor. For non-preemptive scheduling, these job slots are used by running and suspended jobs or by pending jobs that with job slots that are reserved for them. For preemptive scheduling, these job slots are used by running jobs or by pending jobs with slots that are reserved for them. This job limit is configured per processor so that multiprocessor hosts have more job slots. The dash character (-) indicates no limit. JL/P is defined in the LSF configuration file `lsb.users`.

MAX
The maximum number of job slots that can be processed concurrently for the specified users' jobs. For non-preemptive scheduling, these job slots are used by running and suspended jobs or by pending jobs with job slots that are reserved for them. For preemptive scheduling, these job slots are used by running jobs or by pending jobs with job slots that are reserved for them. The dash character (-) indicates no limit. MAX is defined by the **MAX_JOBS** parameter in the `lsb.users` file.

NJOBS
The current number of tasks for all of a users' jobs. A parallel job that is pending is counted as *n* tasks for it uses *n* job slots in the queue when it is dispatched.

If the `-alloc` option is used, total is the sum of the RUN, SSUSP, USUSP, and RSV counters.

PEND
The number of tasks in all of the specified users' pending jobs. If used with the `-alloc` option, the total is `0`.

RUN
The number of tasks in all of the specified users' running jobs. If the `-alloc` option is used, the total is the allocated slots for the users' jobs.

SSUSP
The number of tasks in all of the specified users' system-suspended jobs. If the `-alloc` option is used, total is the allocated slots for the users' jobs.

USUSP
The number of tasks in all of the specified users' user-suspended jobs. If the `-alloc` option is used, total is the allocated slots for the users' jobs.

RSV
The number of tasks that reserve slots for all of the specified users' pending jobs. If the `-alloc` option is used, total is the allocated slots for the users' jobs.

MPEND
The pending job slot threshold for the specified users or user groups. MPEND is defined by the **MAX_PEND_SLOTS** parameter in the `lsb.users` configuration file.

PRIORITY
The APS user (USER) factors for the specified users or user groups. PRIORITY is defined by the **PRIORITY** parameter in the `lsb.users` file. User priority is displayed only with the **busers -w** option.

PJOBS
The number of users' pending jobs.

MPJOBS
The pending job threshold for the specified users. MPJOBS is defined by the **MAX_PEND_JOBS** parameter in the configuration file `lsb.users`.

See also

bugroup, `lsb.users`, `lsb.queues`

Chapter 65. bwait

Pauses and waits for the job query condition to be satisfied.

Synopsis

`bwait -w "`*wait_condition*`" [-t` *timeout*`] [-r` *rereg_interval*`]`

`bwait -h | -V`

Description

The **bwait** command pauses and waits for the specified job condition to occur before the command returns. Use the **bwait** command to reduce workload on the **mbatchd** daemon by including **bwait** in a user script for running jobs instead of using the **bjobs** command in a tight loop to check the job status. For example, the user script might have a command to submit a job, then run **bwait** to wait for the first job to be DONE before continuing the script.

You can also set the maximum amount of time that the **mbatchd** daemon takes to evaluate the wait conditions in a scheduling session by specifying the **EVALUATE_WAIT_CONDITION_TIMEOUT** parameter in the `lsb.params` file. This limits the amount of time that the wait condition evaluation blocks services and frees up time to perform other services during the scheduling cycle.

Options

-w *wait_condition*
Required. Specifies the wait condition to be satisfied. This expression follows the same format as the job dependency expression for the **bsub -w** option. For more details, refer to bsub -w.

-t *timeout*
Optional. Specifies the timeout interval for the wait condition, in minutes. Specify an integer between 1 and 525600 (one year). By default, LSF uses the value of the **DEFAULT_BWAIT_TIMEOUT** parameter in the `lsb.params` file.

-r *rereg_interval*
Optional. Specifies the time interval to re-register the wait condition from the bwait command to the mbatch daemon, in minutes. Specify an integer between 1 and 525600 (one year). By default, LSF uses the value of the **LSB_BWAIT_REREG_INTERVAL** parameter in the `lsf.conf` file.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Chapter 66. ch

Changes the host where subsequent commands run.

Synopsis

`ch [-S] [-t] [host_name]`

`ch [-h | -V]`

Description

The **ch** command has the following default behavior:

- If no arguments are specified, changes the current host to the home host, the host from which the **ch** command was issued.
- Runs commands on the home host.
- Shell mode support is not enabled.
- Does not display run time of tasks.

Use the **ch** command to quickly change to a designated host with the same execution environment. A simple shell is started that delivers all subsequent commands (except built-in commands) to the designated host for execution.

When the simple shell starts, it is in the current working directory and has the same command execution environment as the parent shell. Every remotely dispatched command is run with the same environment as on the home host. The syntax of the **ch** command is similar to the Bourne shell with some important differences.

The ampersand (&) character that follows a command represents a background job in the Bourne shell. The ampersand (&) character is ignored by the **ch** command. You can submit background jobs with the **ch** command with the built-in **post** command and bring them into the foreground with the built-in **contact** command.

The **ch** command recognizes a tilde (~) as a special path name. If a tilde is followed by white space (space, tab, new line or slash (/) character, the ~ character is represents the user's home directory. Otherwise, the ~ represents the home directory of the user name that is given by the string that follows the ~ character. Pipes, lists of commands and redirection of standard input and standard output are all handled by running the **/bin/sh** command.

The following sequence of commands illustrates the behavior of the **ch** command. For example, the user is on `hostA`:

```
ch hostB
hostB> ch hostC
hostC> ch
hostA> ... ...
```

Options

-S
Starts remote tasks with shell mode support. Shell mode support is required for running interactive shells or applications that redefine the `CTRL-C` and `CTRL-Z` keys (for example, **jove**).

-t
Turns on the timing option. The amount of time each subsequent command takes to run is displayed.

host_name
Runs subsequent commands on the specified host.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Usage

The **ch** command interprets the following built-in commands:

cd [*directory_name*]
Changes the current working directory to the specified directory. If a directory is not specified, changes to the user's home directory by default.

ch [*host_name*]
Changes the current working host to the specified host. If a host is not specified, changes to the home host by default.

post [*command* [*argument* ...]]
Posts the specified command for execution in the background on the current working host. The **ch** command assigns a unique task ID to this command and displays the ID. The **ch** command continues to interact with the user. However, the output of background jobs might disturb the screen. You can post multiple commands on one host or on different hosts. When a previously posted command is completed, the **ch** command reports its status to standard error. If a command is not specified, the **ch** command displays all running background commands.

contact *task_ID*
Brings a previously posted background command into the foreground. The task ID (*task_ID*) is the ID returned by the **post** command. Standard input is passed to this foreground command. You cannot put an active foreground job into the background. A command that was brought into the foreground with the **contact** command cannot be put back into the background.

exit
Exits the **ch** command if no posted commands are running. Typing an EOF character (usually `CTRL-D` but can be set otherwise, see **stty**) forces the **ch** command to exit. Posted commands that are not completed are killed.

Limitations

The **ch** command does not support script, history, or alias.

The **ch** command shows the following prompt:
current_working_host`:`*current_working_directory*`>`. If the **ch** session is invoked by a shell that supports job control (such as `tcsh` or `ksh`), `CTRL-Z` suspends the **ch** session. The exit status of a command line is printed to `stderr` if the status is nonzero.

See also

lsrun, **rsh**, **stty**

Chapter 67. gpolicyd

Displays LSF global policy daemon information.

Synopsis

`LSF_SERVERDIR/gpolicyd [-C] [-d dir] [-debug_level]`

`LSF_SERVERDIR/gpolicyd [-h | -V]`

Description

The **gpolicyd** daemon runs on the LSF master and master candidate hosts for global policy management.

A new **gpolicyd** daemon is started if the host for the current **gpolicyd** daemon fails.

The **gpolicyd** daemon reads the `lsf.conf` file to get the environment information. Use the `-d` option to change the default location of the LSF environment directory that contains the `lsf.conf` file.

Options

-C

The **gpolicyd** daemon checks the syntax of the LSF configuration files, prints verbose messages to `stdout`, and exits. The **gpolicyd** daemon with the `-C` option does not to run on the master host.

Attention: Do not start the **gpolicyd** daemon manually without the `-C` option.

-d env_dir

Reads the `lsf.conf` file from the directory that is specified by *env_dir*, rather than from the default directory `/etc`, or from the directory set by the **LSF_ENVDIR** parameter.

-debug_level

Sets debug mode level. Valid values are either 1 or 2. When debug mode is set, the daemons are run in debug mode and can be started by a normal user (non-root). If debug level is 1, the **gpolicyd** daemon runs in the background when started. If debug level is 2, the **gpolicyd** daemon stays in the foreground. Therefore, the daemon is always started by the daemon with the same options, and has**gpolicydsbatchd** the same debug level. If LSF daemons are running in debug mode, the **LSB_DEBUG** parameter must be defined in the `lsf.conf` file for LSF commands to communicate with the daemons.

ERROR REPORTING

The **gpolicyd** daemon has no controlling TTY. Errors are sent to `syslog` with log level **LOG_ERR**, or, if the **LSF_LOGDIR** parameter is defined in the `lsf.conf` file, which is written to the file `LSF_LOGDIR/gpolicyd.log.host_name`.

FILES

`lsb.globalpolicies`

Chapter 68. lim

Load information manager (LIM) daemon or service, monitoring host load.

Synopsis

`lim [-C] [-t] [-T] [-vm] [-d conf_dir] [-debug_level]`

`lim -h`

`lim -V`

Description

There is one **lim** daemon or service on every host in the cluster. Of these, one **lim** from the master list is elected master **lim** for the cluster. The master **lim** receives load information from the other **lim** daemons, and provides services to all host.

The lim does the following for the host on which it runs:

- Starts **pem** on that host
- Provides system configuration information to **vemkd**
- Monitors load and provides load information statistics to **vemkd** and users

The master lim starts **vemkd** and **pem** on the master host.

The non-master lim daemons monitor the status of the master **lim** and elect a new master (from the master list) if the current master **lim** becomes unavailable.

Collectively, the LIMs in the cluster coordinate the collection and transmission of load information. Load information is collected in the form of load indices.

CAUTION: Never start the daemon manually without options: specify the **-V** option to check the version, the **-d** option to start the daemon in debug mode, or the **-C** option to validate its configuration files.

Options

-d conf_dir
Starts the daemon, reading from the LSF configuration file `ego.conf` in the specified directory, rather than from the directory set via the EGO_CONFDIR environment variable.

Use this option when starting the daemon in debug mode.

CAUTION: Never start the daemon manually unless directed to do so by Product Support.

-debug_level
Starts the lim in debug mode. When running in debug mode, the lim uses a hard-coded port number rather than the one registered in system services.

Specify one of the following values:

-1
Starts the lim in the background, with no associated control terminal.

-2
Starts the lim in the foreground, displaying the log messages to the terminal.

CAUTION: Never start the daemon manually unless directed to do so by Product Support.

-t

Displays host information, such as host type, host architecture, number of physical processors, number of cores per physical processor, number of threads per core, and license requirements.

Note: When running Linux kernel version 2.4, you must run **lim -t** as root to ensure consistent output with other clustered application management commands (for example, output from running theLSF command **lshosts**).

-T

Displays host topology information for each host or cluster. Topology is displayed by *processor unit* level: NUMA node, if present, socket, core, and thread,

A socket is a collection of cores with a direct pipe to memory. Each socket contains 1 or more cores. This does not necessarily refer to a physical socket, but rather to the memory architecture of the machine.

A core is a single entity capable of performing computations.

A node contains sockets, a socket contains cores, and a core can contain threads if the core is enabled for multithreading.

The following fields are displayed:

Host[*memory*] *host_name*

Maximum memory available on the host followed by the host name. If memory availability cannot be determined, a dash (-) is displayed for the host.

For hosts that do not support affinity scheduling, a dash (-) is displayed for host memory and no host topology is displayed.

NUMA[*numa_node*: *max_mem*]

Maximum NUMA node memory. It is possible for requested memory for the NUMA node to be greater than the maximum available memory displayed.

If no NUMA nodes are present, then the NUMA layer in the output is not shown. Other relevant items such as host, socket, core and thread are still shown.

If the host is not available, only the host name is displayed. A dash (-) is shown where available host memory would normally be displayed.

In the following example, full topology (NUMA, socket, and core) information is shown for hostA:

```
lim -T
Host[24G] hostA
    NUMA[0: 24G]
        Socket
            core(0)
            core(1)
            core(2)
            core(3)
        Socket
            core(4)
            core(5)
            core(6)
            core(7)
```

Host hostB has a different architecture:

```
lim -T
Host[63G] hostB
    Socket
        NUMA[0: 16G]
            core(0)
            core(2)
            core(4)
            core(6)
        NUMA[1: 16G]
            core(8)
            core(10)
            core(12)
            core(14)
```

```
    Socket
        NUMA[2: 16G]
            core(1)
            core(3)
            core(5)
            core(7)
        NUMA[3: 16G]
            core(9)
            core(11)
            core(13)
            core(15)
```

When LSF cannot detect processor unit topology, it displays processor units to the closest level. For example:

```
lim -T
     Host[1009M] hostA
            Socket (0 1)
```

On `hostA` there are two processor units: 0 and 1. LSF cannot detect core information, so the processor unit is attached to the socket level.

-vm

-h

Outputs command usage and exits.

-V

Outputs product version and exits.

Files

ego.conf

The lim reads the configuration file `ego.conf` to retrieve configuration information. `ego.conf` is a generic configuration file shared by all daemons/services and clients. It contains configuration information and other information that dictates the behavior of the software.

Some of the parameters lim retrieves from `ego.conf` are as follows:

EGO_LIM_PORT

The TCP port the lim uses to serve all applications.

EGO_SERVERDIR

The directory used for reconfiguring the LIM—where the **lim** binary is stored.

EGO_LOGDIR

The directory used for message logs.

EGO_LOG_MASK

The log level used to determine the amount of detail logged.

EGO_DEBUG_LIM

The log class setting for **lim**.

EGO_ENTITLEMENT_FILE

The full path to and name of the entitlement file.

EGO_DEFINE_NCPUS

Defines whether ncpus is to be defined as procs, cores, or threads. This parameter overrides LSF_ENABLE_DUALCORE. If EGO_ENABLE_DUALCORE is set, EGO_DEFINE_NCPUS settings take precedent.

- procs (if ncpus defined as procs, then ncpus = nprocs)cores (if ncpus defined as cores, then ncpus = nprocs x ncores)
- threads (if ncpus defined as threads, then ncpus = nprocs x ncores x nthreads)

Note:

When EGO_DEFINE_NCPUS is set, run queue-length values (r1* values returned by lsload) are automatically normalized based on the set value.

If EGO_DEFINE_NCPUS is not defined, but EGO_ENABLE_DUALCORE is set, the lim reports the number of cores. If both EGO_DEFINE_NCPUS and LSF_ENABLE_DUALCORE are set, then the EGO parameter takes precedence.

EGO_ENABLE_DUALCORE
Defines if the hosts have dual cores or not. Is overridden by EGO_DEFINE_NCPUS, if set.

Note: If EGO_DEFINE_NCPUS is not defined, but EGO_ENABLE_DUALCORE is set, the lim reports the number of cores. If both EGO_DEFINE_NCPUS and LSF_ENABLE_DUALCORE are set, then the EGO parameter takes precedence.

Customization

You can customize the **lim** by changing configuration files in EGO_CONFDIR directory. Configure `ego.cluster.<cluster_name>` to define various cluster properties such as the resources on individual hosts, the load threshold values for a host, and so on. Configure `ego.shared` to define host models read by the **lim**, or the CPU factor of individual hosts.

Chapter 69. lsacct

Displays accounting statistics on finished RES tasks in the LSF system.

Synopsis

`lsacct` [`-l`] [`-C` *time0*,*time1*] [`-S` *time0*,*time1*] [`-f` *logfile_name*] [`-m` *host_name*] [`-u` *user_name* ... | `-u all`] [*pid* ...]

`lsacct` [`-h` | `-V`]

Description

Displays statistics on finished tasks that are run through RES. When a remote task completes, RES logs task statistics in the task log file.

By default, displays accounting statistics for only tasks that are owned by the user who ran the **lsacct** command.

By default, displays accounting statistics for tasks that are executed on all hosts in the LSF system.

By default, displays statistics for tasks that are logged in the task log file that is currently used by RES: `LSF_RES_ACCTDIR/lsf.acct.`*host_name* or `/tmp/lsf.acct.`*host_name*.

If the `-l` option is not specified, the default is to display only the fields in SUMMARY.

The RES on each host writes its own accounting log file. These files can be merged by using the **lsacctmrg** command to generate statistics for the entire LSF cluster.

All times are reported in seconds. All sizes are reported in KB.

Options

-l
Per-task statistics. Displays statistics about each task.

-C *time0*,*time1*
Displays accounting statistics for only tasks that completed or exited during the specified time interval.

The time format is the same as in the **bhist** command.

-f *logfile_name*
Searches the specified task log file for accounting statistics. Specify either an absolute or a relative path.

Useful for analyzing old task log files or files that are merged with the `lsacctmrg` command.

-m *host_name* ...
Displays accounting statistics only for tasks that are executed on the specified hosts.

If a list of hosts is specified, host names must be separated by spaces and enclosed in quotation marks (") or (').

-S *time0*,*time1*
Displays accounting statistics only for tasks that began running during the specified time interval.

The time format is the same as in the **bhist** command.

-u *user_name* ... | -u all
Displays accounting statistics only for tasks that are owned by the specified users, or by all users if the keyword `all` is specified.

If a list of users is specified, user names must be separated by spaces and enclosed in quotation marks (") or ('). You can specify both user names and user IDs in the list of users. To specify a

Windows user account, include the domain name in uppercase letters and use a single backslash (*DOMAIN_NAME**user_name*) on a Windows command prompt or a double backslash (*DOMAIN_NAME* *user_name*) on a UNIX or Linux command line.

***pid* ...**
Displays accounting statistics only for tasks with the specified process ID. This option overrides all other options except for the `-l`, `-f`, `-h`, and `-V` options.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Default output

Overall statistics for tasks are displayed. The SUMMARY is the default output format.

The total, average, maximum, and minimum resource usage statistics apply to all specified tasks.

The following fields are displayed:

Total number of tasks
Total number of tasks, including tasks that are completed successfully and total number of exited tasks.

Time range of started tasks
Start time of the first and last task selected.

Time range of ended tasks
Completion or exit time of the first and last task selected.

Resource usage of tasks selected
See **getrusage**.

CPU time
Total CPU time that is consumed by the task.

Page faults
Number of page faults.

Swaps
Number of times the process was swapped out.

Blocks in
Number of input blocks.

Blocks out
Number of output blocks.

Messages sent
Number of System VIPC messages sent.

Messages rcvd
Number of IPC messages received.

Voluntary cont sw
Number of voluntary context switches.

Involuntary con sw
Number of involuntary context switches.

Turnaround
Elapsed time from task execution to task completion.

Per-task statistics with the -l option

In addition to the fields displayed by default in SUMMARY, displays the following fields for each task:

Starting time

Time the task started.

User and host name

User who submitted the task and the host from which the task was submitted, in the format *user_name*@*host*.

PID

UNIX process ID of the task.

Execution host

Host on which the command was run.

Command line
Complete command line that was run.

CWD
Current working directory of the task.

Completion time
Time at which the task completed.

Exit status
UNIX exit status of the task.

Files

Reads the `lsf.acct.`*`host_name`* file.

See also

bhist, **lsacctmrg**, **res**, `lsf.acct`

Chapter 70. lsacctmrg

Merges LSF RES task log files.

Synopsis

`lsacctmrg` [`-f`] *logfile_name ... target_logfile_name*

`lsacctmrg` [`-h` | `-V`]

Description

Merges specified task log files into the specified target file in chronological order according to completion time.

All files must be in the format that is specified in the `lsf.acct` file.

Options

-f
Overwrites the target file without prompting for confirmation.

logfile_name ...
Specify log files to be merged into the target file, separated by spaces. Specify either an absolute or a relative path.

target_logfile_name
Specify the file into which all log files are to be merged. Specify either an absolute or a relative path. The target file cannot be part of the files to be merged.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

`lsf.acct`, **res**

Chapter 71. lsadmin

Administrative tool to control LIM and RES daemon operations in LSF.

Synopsis

`lsadmin` *subcommand*

`lsadmin` [`-h` | `-V`]

Description

Important: This command can be used only by LSF administrators.

The **lsadmin** command runs privileged subcommands to control LIM and RES daemon operations in the LSF cluster.

If you do not include subcommands, the `lsadmin` command prompts for subcommands from the standard input.

When you use subcommands for which multiple host names or host groups can be specified, do not enclose the multiple names in quotation marks.

When live configuration with the **bconf** command is enabled (the **LSF_LIVE_CONFDIR** parameter is defined in the `lsf.conf` file), the **lsadmin** command uses configuration files that are generated by the **bconf** command.

Subcommand synopsis

`ckconfig` [`-v`]

`reconfig` [`-f`] [`-v`]

`limstartup` [`-f`] [*host_name* ... |`all`]

`limshutdown` [`-f`] [*host_name* ... | `all`]

`limrestart` [`-v`] [`-f`] [*host_name* ... | `all`]

`limlock` [`-l` *time_seconds*]

`limunlock`

`resstartup` [`-f`] [*host_name* ... | `all`]

`resshutdown` [`-f`] [*host_name* ... | `all`]

`resrestart` [`-f`] [*host_name* ... | `all`]

`reslogon` [`-c` *cpu_time*] [*host_name* ... | `all`]

`reslogoff` [*host_name* ... | `all`]

`limdebug` [`-c` *class_name* ...] [`-l` *debug_level*] [`-f` *logfile_name*] [`-o`] [*host_name* ...]

`resdebug` [`-c` *class_name*] [`-l` *debug_level*] [`-f` *logfile_name*] [`-o`] [*host_name* ...]

`limtime` [`-l` *timing_level*] [`-f` *logfile_name*] [`-o`] [*host_name* ...]

`restime` [`-l` *timing_level*] [`-f` *logfile_name*] [`-o`] [*host_name* ...]

`showconf lim` [*host_name* ... | `all`]

`help` [*subcommand* ...] | `?` [*subcommand* ...]

`quit`

`-h`

`-V`

Options

subcommand
Runs the specified subcommand. See the Usage section.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Usage

ckconfig [-v]
Checks LSF configuration files.

-v
Displays detailed messages about configuration file checking.

reconfig [-f] [-v]
Restarts LIM on all hosts in the cluster. Use the **reconfig** subcommand after you change configuration files. The configuration files are checked before all LIMs in the cluster are restarted. If the configuration files are not correct, reconfiguration is not started.

If the **LSF_MASTER_LIST** parameter is specified in the `lsf.conf` file, you are prompted to confirm the reconfiguration for only the master candidate hosts.

-f
Disables user interaction and forces LIM to restart on all hosts in the cluster if no unrecoverable errors are found. This option is useful in batch mode.

-v
Displays detailed messages about configuration file checking.

limstartup [-f] [*host_name* ... |all]
Starts LIM on the local host if no arguments are specified.

Starts LIMs on the specified hosts or on all hosts in the cluster if the word `all` is the only argument provided. You are prompted to confirm LIM startup.

Only root and users who are listed in the parameter **LSF_STARTUP_USERS** in the `lsf.sudoers` file can use the `all` and `-f` options to start LIM as root.

If permission to start LIMs as root is not configured, use the **limstartup** subcommand to start LIMs as yourself after your confirmation.

-f
Disables interaction and does not ask for confirmation for starting LIMs.

limshutdown [-f] [*host_name* ... | all]
Shuts down LIM on the local host if no arguments are supplied.

Shuts down LIMs on the specified hosts or on all hosts in the cluster if the word `all` is specified. You are prompted to confirm LIM shutdown.

-f

Disables interaction and does not ask for confirmation for shutting down LIMs.

limrestart [-v] [-f] [*host_name* ... | all]

Restarts LIM on the local host if no arguments are supplied.

Restarts LIMs on the specified hosts or on all hosts in the cluster if the word `all` is specified. You are prompted to confirm LIM restart.

Use the **limrestart** subcommand with care. Do not change the cluster until all the LIMs start. If you run the **limrestart** *host_name* ... subcommand to restart some of the LIMs after you change the configuration files, but other LIMs are still running the old configuration, confusion arises among these LIMs. To avoid this situation, use the **reconfig** subcommand instead of the **limrestart** subcommand.

-v
Displays detailed messages about configuration file checking.

-f
Disables user interaction and forces LIM to restart if no unrecoverable errors are found. This option is useful in batch mode. The **limrestart -f all** subcommand is the same as the **reconfig -f** subcommand.

limlock [-l *time_seconds*]
Locks LIM on the local host until it is explicitly unlocked if no time is specified. When a host is locked, LIM's load status becomes lockU. No job is sent to a locked host by LSF.

-l time_seconds
The host is locked for the specified time in seconds.

LSF suspends all non-exclusive jobs that are running on the host. Locking a host is useful if it is running an exclusive job that required all the available CPU time, memory, or both. If the **LSB_DISABLE_LIMLOCK_EXCL=y** parameter is set, to enable preemption of exclusive jobs, for example, LSF suspends all jobs, including exclusive jobs.

limunlock
Unlocks LIM on the local host.

resstartup [-f] [*host_name* ... | all]
Starts RES on the local host if no arguments are specified.

Starts RES on the specified hosts or on all hosts in the cluster if the word all is specified. You are prompted to confirm RES startup.

Only root and users who are defined by the **LSF_STARTUP_USERS** parameter in the lsf.sudoers file can use the all and -f options to start RES as root.

For root installation to work properly, the **lsadmin** command must be installed as a setuid to root program.

-f
Disables interaction and does not ask for confirmation for starting RESs.

resshutdown [-f] [*host_name* ... | all]
Shuts down RES on the local host if no arguments are specified.

Shuts down RES on the specified hosts or on all hosts in the cluster if the word all is specified. You are prompted to confirm RES shutdown.

If RES is running, it keeps running until all remote tasks exit.

-f
Disables interaction and does not ask for confirmation for shutting down RES.

resrestart [-f] [*host_name* ... | all]
Restarts RES on the local host if no arguments are specified.

Restarts RES on the specified hosts or on all hosts in the cluster if the word all is specified. You are prompted to confirm RES restart.

If RES is running, it keeps running until all remote tasks exit. While the **resrestart** subcommand waits for remote tasks to exit, another RES is restarted to serve the new queries.

-f
Disables interaction and does not ask for confirmation for restarting RES.

reslogon [-c *cpu_time*] [*host_name* ... | all]
Logs all tasks that are run by RES on the local host if no arguments are specified.

Logs tasks that are run by RES on the specified hosts or on all hosts in the cluster if `all` is specified.

RES writes the task resource usage information into the log file `lsf.acct.`*`host_name`*. The location of the log file is determined by the **LSF_RES_ACCTDIR** parameter in the `lsf.conf` file. If the **LSF_RES_ACCTDIR** parameter is not defined, or RES cannot access it, the log file is created in the `/tmp` directory instead.

-c *cpu_time*
Logs only tasks that use more than the specified amount of CPU time. The amount of CPU time is specified by *cpu_time* in milliseconds.

reslogoff [*host_name* ... | all]
Turns off RES task logging on the specified hosts or on all hosts in the cluster if `all` is specified.

If no arguments are specified, turns off RES task logging on the local host.

limdebug [-c *class_name* ...] [-l *debug_level*] [-f *logfile_name*] [-o] [*host_name* ...]
Sets the message log level for LIM to include additional information in log files. You must be root or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

***class_name*=0**
No additional classes are logged.

***debug_level*=0**
LOG_DEBUG level in parameter **LSF_LOG_MASK**.

logfile_namedaemon_name*.log.*host_name
LSF system log file in the LSF system log file directory, in the format *daemon_name*.log.*host_name*.

host_name*=*local_host
Host from which the command was submitted.

In LSF multicluster capability, debug levels can be set only for hosts within the same cluster. For example, you cannot set debug or timing levels from a host in `clusterA` for a host in `clusterB`. You need to be on a host in `clusterB` to set up debug or timing levels for `clusterB` hosts.

-c *class_name* ...
Specify software classes for which debug messages are to be logged.

By default, no additional classes are logged (class name `0`).

Note: Classes are also listed in the `lsf.h` header file.

The following classes are supported:

LC_AFS and LC2_AFS
Log AFS messages.

LC_AUTH and LC2_AUTH
Log authentication messages.

LC_CHKPNT
Log checkpointing messages.

LC_COMM and LC2_COMM
Log communication messages.

LC_CONF
Print all parameters in the `lsf.conf` and `ego.conf` files.

LC_DCE and LC2_DCE
Log messages that pertain to DCE support.

LC_EXEC and LC2_EXEC
Log significant steps for job execution.

LC_FILE and LC2_FILE
Log file transfer messages.

LC_HANG and LC2_HANG
Mark where a program might hang.

LC_MULTI and LC2_MULTI
Log messages that pertain to LSF multicluster capability.

LC_PIM and LC2_PIM
Log PIM messages.

LC_SIGNAL and LC2_SIGNAL
Log messages that pertain to signals.

LC_TRACE and LC2_TRACE
Log significant program walk steps.

LC_XDR and LC2_XDR
Log everything that is transferred by XDR.

-l ***debug_level***
Specify level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower levels.

The default is `0` (`LOG_DEBUG` level in parameter **LSF_LOG_MASK**)

The following values are supported:

0
`LOG_DEBUG` level for parameter **LSF_LOG_MASK** in the `lsf.conf` file. `0` is the default.

1
`LOG_DEBUG1` level for extended logging. A higher level includes lower logging levels. For example, `LOG_DEBUG1` includes `LOG_DEBUG` levels.

2
`LOG_DEBUG2` level for extended logging. A higher level includes lower logging levels. For example, `LOG_DEBUG2` includes `LOG_DEBUG1` and `LOG_DEBUG` levels.

3
`LOG_DEBUG3` level for extended logging. A higher level includes lower logging levels. For example, `LOG_DEBUG3` includes `LOG_DEBUG2`, `LOG_DEBUG1`, and `LOG_DEBUG` levels.

-f ***logfile_name***
Specify the name of the file into which debugging messages are to be logged. You can specify a file name with or without a full path.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file that is created has the following format:

```
logfile_name.daemon_name.log.host_name
```

On UNIX and Linux, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, if the specified path is not valid, no log file is created.

By default, current LSF system log file in the LSF system log file directory is used.

-o
Turns off temporary debug settings and resets them to the daemon start state. The message log level is reset back to the value of **LSF_LOG_MASK** and classes are reset to the value of **LSB_DEBUG_RES**, **LSB_DEBUG_LIM**.

The log file is also reset back to the default log file.

host_name ...
Optional. Sets debug settings on the specified host or hosts.

The default is the local host (the host from which command was submitted).

resdebug [-c *class_name*] [-l *debug_level*] [-f *logfile_name*] [-o] [*host_name* ...]

Sets the message log level for RES to include additional information in log files. You must be the LSF administrator to use this command, not root.

See description of **limdebug** for an explanation of options.

limtime [-l *timing_level*] [-f *logfile_name*] [-o] [*host_name* ...]

Sets timing level for the LIM daemon to include extra timing information in log files. You must be root or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

***timing_level*=no**
Timing information is recorded.

***logfile_name*=current**
LSF system log file in the LSF system log file directory, in the format *daemon_name*.log.*host_name*.

***host_name*=local**
The host from which command was submitted.

In LSF multicluster capability, timing levels can be set only for hosts within the same cluster. For example, you cannot set debug or timing levels from a host in clusterA for a host in clusterB. You need to be on a host in clusterB to set up debug or timing levels for clusterB hosts.

-l *timing_level*
Specifies detail of timing information that is included in log files. Timing messages indicate the execution time of functions in the software and are logged in milliseconds.

The following values are supported: 1 | 2 | 3 | 4 | 5

The higher the number, the more functions in the software that are timed and whose execution time is logged. The lower numbers include more common software functions. Higher levels include all lower levels.

By default, no timing information is logged.

-f *logfile_name*
Specify the name of the file into which timing messages are to be logged. You can specify a file name with or without a full path.

If a file name without a path is specified, the file is saved in the LSF system log file directory.

The name of the file that is created has the following format:

```
logfile_name.daemon_name.log.host_name
```

On UNIX and Linux, if the specified path is not valid, the log file is created in the /tmp directory.

On Windows, if the specified path is not valid, no log file is created.

Note: Both timing and debug messages are logged in the same files.

The default is the current LSF system log file in the LSF system log file directory, in the format *daemon_name*.log.*host_name*.

-o
Optional. Turn off temporary timing settings and reset them to the daemon start state. The timing level is reset back to the value of the parameter for the corresponding daemon (**LSB_TIME_LIM**, **LSB_TIME_RES**).

The log file is also reset back to the default log file.

***host_name* ...**

Sets the timing level on the specified host or hosts.

By default, timing level is set on the local host (host from which command was submitted).

restime [-l *timing_level*] [-f *logfile_name*] [-o] [*host_name* ...]

Sets timing level for RES to include extra timing information in log files. You must be the LSF administrator to use this command, not root.

See description of `limtime` for an explanation of options.

showconf lim [*host_name* ... | all]

Displays all configured parameters and their values set in the `lsf.conf` or `ego.conf` file that affect the **lim** daemon.

By default, the **lsadmin** command displays the local LIM parameters. You can optionally specify the host to display the LIM parameters.

In LSF multicluster capability, the **lsadmin showconf** command displays only the parameters of daemons on the local cluster.

Running the **lsadmin showconf** command from a master candidate host reaches all server hosts in the cluster. Running the **lsadmin showconf** command from a slave-only host might not be able to reach other slave-only hosts.

You cannot run the **lsadmin showconf lim** command from client hosts. **lsadmin** shows only server host configuration, not client host configuration.

The **lsadmin showconf** command displays only the values that are used by LSF.

The LIM daemon reads the **EGO_MASTER_LIST** parameter from wherever it is defined. You can define either the **LSF_MASTER_LIST** parameter in the `lsf.conf` file or the **EGO_MASTER_LIST** parameter in the `ego.conf` file. If EGO is enabled in the LSF cluster, LIM reads the `lsf.conf` file first, and then the `ego.conf` file. LIM takes only the value of the **LSF_MASTER_LIST** parameter if the **EGO_MASTER_LIST** parameter is not defined at all in the `ego.conf` file.

For example, if you define the **LSF_MASTER_LIST** parameter in the `lsf.conf`file, and the **EGO_MASTER_LIST** parameter in the `ego.conf` file, the **lsadmin showconf** command displays the value of the **EGO_MASTER_LIST** parameter.

If EGO is disabled, the `ego.conf` file is not loaded, so whatever is defined in the `lsf.conf` file is displayed.

help [*subcommand* ...] | ? [*subcommand* ...]

Displays the syntax and functions of the specified subcommands.

From the command prompt, you can use **help** or **?**.

quit

Exits the **lsadmin** session.

See also

bmgroup, **busers**, `lsf.conf`, `lsf.sudoers`, `lsf.acct`

Chapter 72. lsclusters

Displays configuration information about LSF clusters.

Synopsis

`lsclusters` [`-h`] [`-V`] [`-l` | `-w`] [*cluster_name* ...]

Description

By default, returns information about the local cluster and all other clusters that the local cluster is aware of. The local cluster detects all clusters that are defined in the `RemoteClusters` section of the `lsf.cluster.`*`cluster_name`* file if that section exists, or all clusters defined in the `lsf.shared` file.

Options

-l
Long format. Displays additional information.

-w
Wide format. Displays additional information. Fields are displayed without truncation.

***cluster_name* ...**
Displays information about the specified clusters.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Default output

A listing of the clusters is displayed with the following fields:

CLUSTER_NAME
The name of the cluster.

STATUS
The status of the cluster. The following values are supported:

ok
The cluster is in normal load sharing state, and can exchange load information with the local cluster.

unavail
The cluster is unavailable.

MASTER_HOST
The name of the cluster's master host.

ADMIN
The user account name of the cluster's primary LSF administrator.

HOSTS

Number of LSF static client and server hosts in the cluster. The HOSTS field does not include floating clients.

SERVERS
Number of LSF server hosts in the cluster.

Long format with the -l option

The `-l` lists cluster administrator login names, available host-based resource names, host types, host models, whether the local cluster accepts or sends interactive jobs to this cluster, preferred authentication name, and the actual authentication name in use.

See also

`ls_info`, `ls_policy`, `ls_clusterinfo` `lsf.cluster`

Chapter 73. lseligible

Displays whether a task is eligible for remote execution.

Synopsis

`lseligible` [`-r`] [`-q`] [`-s`] *task_name*

`lseligible` [`-h` | `-V`]

Description

By default, only tasks in the remote task list are considered eligible for remote execution.

Options

-q
Quiet mode. Displays only the resource requirement string that is defined for the task. The string `ELIGIBLE` or `NON-ELIGIBLE` is omitted.

-r
Remote mode. Considers eligible for remote execution any task not included in the local task list.

-s
Silent mode. No output is produced. The `-q` and `-s` options are useful for shell scripts that operate by testing the exit status.

task_name
Specify a command.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Output

If the task is eligible, the string `ELIGIBLE` followed by the resource requirements that are associated with the task are printed to `stdout`. Otherwise, the string `NON-ELIGIBLE` is printed to stdout.

If the **lseligible** command prints `ELIGIBLE` with no resource requirements, the task has the default requirements of CPU consumption and memory usage.

Diagnostics

The **lseligible** command has the following exit statuses:

0
Task is eligible for remote execution.

1
Command is to be run locally.

-1
Syntax errors.

-10
A failure is detected in the LSF system.

See also

`ls_eligible`, **lsrtasks**, `lsf.tasklsf.task`

Chapter 74. lsfinstall

The LSF installation and configuration script.

Synopsis

```
lsfinstall -f install.config
lsfinstall -s -f slave.config
lsfinstall -h
```

Description

The **lsfinstall** script runs the LSF installation scripts and configuration utilities to install a new LSF cluster or upgrade LSF from a previous release.

To install a fully operational LSF cluster that all users can access, run the **lsfinstall** script as root.

You can run the **lsfinstall** script as a non-root user, with limitations.

Required install.config variables

The following parameters are required for installation with the `install.config` configuration file:

- `LSF_TOP="/path"`
- `LSF_ADMINS="user_name [user_name ...]"`
- `LSF_CLUSTER_NAME="cluster_name"`

Required slave.config variables

If you use the `slave.config` file to install dynamic slave hosts, the following parameters are required:

- `LSF_TOP="/path"`
- `LSF_TARDIR="/path"`
- `LSF_SERVER_HOSTS="host_name [host_name ...]"`

Variables that require an absolute path

- `LSF_TOP="/path"`
- `LSF_TARDIR="/path"`
- `LSF_ENTITLEMENT_FILE="/path"`

What lsfinstall does

Before it installs and configures LSF, the **lsfinstall** script checks the installation prerequisites, and outputs the results to the `lsfprechk.rpt` file. The **lsfinstall** script writes any unrecoverable errors to the `Install.err` file and exits. You must correct these errors before you continue to install and configure LSF.

During installation, the **lsfinstall** script logs installation progress in the `Install.log` file and calls other utilities to decompress, extract, and copy product files. After it copies files to the installation location, the script configures the cluster.

(Optional) run hostsetup

After installation, you can run the **hostsetup** command to set up LSF hosts and automatic LSF startup. After you set up the server hosts, start your cluster and test the installation by running some basic commands.

Run the **hostsetup** command on each LSF server host.

For complete usage of the **hostsetup** command, run the **hostsetup -h** command to see the command usage and options help.

Note: Running the **hostsetup** command is only required if you plan to run IBM POE jobs by using IBM Parallel Environment (IBM PE).

Important: Before you run the **hostsetup** command, make sure that the hosts you want to set up are in the `lsf.cluster.`*`cluster_name`* file.

For example, run the following commands to use the LSF cluster that is installed in the `/usr/share/lsf` directory and configure LSF daemons to start automatically at system startup time:

```
# cd /usr/share/lsf/10.1/install
# ./hostsetup --top="/usr/share/lsf" --boot="y"
```

Where lsfinstall is located

lsfinstall is included in the LSF installation script TAR file `lsf10.1.0.9_lsfinstall.tar.Z` and is located in the `lsf10.1.0.9_lsfinstall` directory that is created when you decompress and extract installation script TAR file.

After you install LSF, the **lsfinstall** script is located in the `LSF_TOP/10.1/install/` directory.

Options

-f ***option_file***
Name of the file that specifies the installation options. The file can be any name that you choose. The name of the default template file for normal installation is `install.config`. To install slave hosts for dynamic host configuration, use the template file `slave.config`.

-s
Install a dynamic slave host.

Specify installation options in the `slave.config` file.

The following parameters are required:

- `LSF_SERVER_HOSTS="`*`host_name`* `[`*`host_name ...`*`]"`
- `LSF_TOP="/`*`path`*`"`
- `LSF_TARDIR="/`*`path`*`"`

The following parameters are optional:

LSF_LIM_PORT=*port_number*
If the master host does not use the default LSF_LIM_PORT, you must specify the same LSF_LIM_PORT defined in `lsf.conf` on the master host.

LSF_LOCAL_RESOURCES="*resource ...*"
Defines the local resources for a dynamic host.

- For numeric resources, use name-value pairs:

  ```
  "[resourcemap value*resource_name]"
  ```

- For Boolean resources, define resource names:

  ```
  "[resource resource_name]"
  ```

The following example defines a numeric resource for `verilog` licenses and Boolean resource `linux`:

```
LSF_LOCAL_RESOURCES="[hostname hostA] [server 1] [resourcemap 1*verilog] [resource
linux]"
```

Tip: If **LSF_LOCAL_RESOURCES** are already defined in a local `lsf.conf` file on the slave host, the **lsfinstall** script does not add resources that you defined in **LSF_LOCAL_RESOURCES** in the `slave.config` file.

The **lsfinstall** script creates a local `lsf.conf` file for the slave host, which sets the following parameters:

- `LSF_CONFDIR="/path"`
- `LSF_GET_CONF=lim`
- `LSF_LIM_PORT=port_number`
- `LSF_LOCAL_RESOURCES="resource ..."`
- `LSF_SERVER_HOSTS="host_name [host_name ...]"`
- `LSF_VERSION=10.1`

-h

Prints command usage and exits.

See also

`lsf.conf`, `install.configinstall.config`, `slave.config`

Chapter 75. lsfmon

Install or uninstall LSF Monitor in an existing cluster.

Synopsis

```
lsfmon -install
```

```
lsfmon -remove
```

Description

LSF Monitor runs on Microsoft Windows and enables Windows Performance Monitor to chart information about the LSF cluster.

The LSF Monitor service runs under the account of an LSF cluster administrator.

Options

-install
Install LSF Monitor on the host.

-remove
Remove LSF Monitor from the host.

Chapter 76. lsfrestart

Restarts the LIM, RES, **sbatchd**, and **mbatchd** daemons on all hosts in the cluster

Synopsis

`lsfrestart [-f | -pdsh]`

`lsfrestart [-h | -V]`

Description

Important:

This command can be used only by root, the primary cluster administrator, or users who are listed in the `lsf.sudoers` file.

Restarts the LIM, RES, **sbatchd** and **mbatchd** daemons, in that order, on all hosts in the local cluster. When live configuration with the **bconf** command is enabled (the **LSF_LIVE_CONFDIR** parameter is defined in the `lsf.conf` file), the **lsfstartup** command uses configuration files that are generated by the **bconf** command.

By default, prompts for confirmation of the next operation if an error is encountered.

To be able to control all daemons in the cluster, the file `/etc/lsf.sudoers` must be set up properly.

Options

-f
Force mode. Continues to restart daemons even if an error is encountered.

-pdsh
Enable parallel remote command execution with the PDSH tool. The PDSH tool is required on master hosts and master candidates. The chunk size is 400 hosts.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

lsadmin, **badmin**, **lsfshutdown**, **lsfshutdown**, `lsf.sudoers`lsf.sudoers

Chapter 77. lsfshutdown

Shuts down the LIM, RES, **sbatchd**, and **mbatchd** daemons on all hosts in the cluster.

Synopsis

`lsfshutdown` [`-f` | `-pdsh`]

`lsfshutdown` [`-h` | `-V`]

Description

Important:

This command can be used only by root, the primary cluster administrator, or users who are listed in the `lsf.sudoers` file.

Shuts down the **sbatchd**, RES, LIM, and **mbatchd** daemons, in that order, on all hosts.

By default, prompts for confirmation of the next operation if an error is encountered.

To be able to control all daemons in the cluster, the file `/etc/lsf.sudoers` must be set up properly.

Options

-f
Force mode. Continues to shut down daemons even if an error is encountered.

-pdsh
Enable parallel remote command execution with the PDSH tool. The PDSH tool is required on master hosts and master candidates. The chunk size is 400 hosts.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

lsadmin, **badmin**, **lsfrestart**, **lsfstartup**, `lsf.sudoers`lsf.sudoers

Chapter 78. lsfstartup

Starts the LIM, RES, and **sbatchd** daemons on all hosts in the cluster.

Synopsis

`lsfstartup -pdsh [-delay` *seconds*`] [-num_hosts` *number*`]`

`lsfstartup [-f]`

`lsfstartup [-h | -V]`

Description

Important: This command can be used only by root or users who are listed in the `lsf.sudoers` file.

Starts the LIM, RES, and **sbatchd** daemons, in that order, on all hosts. When live configuration with the **bconf** command is enabled (the **LSF_LIVE_CONFDIR** parameter is defined in the `lsf.conf` file), the **lsfstartup** command uses configuration files that are generated by the **bconf** command.

By default, prompts for confirmation of the next operation if an error is encountered.

If LSF daemons are already running, use the **lsfrestart** command instead, or use the **lsfshutdown** to shut down the running daemons before you use the **lsfstartup** command.

To be able to control all daemons in the cluster, the file `/etc/lsf.sudoers` must be set up properly.

Options

-f
Force mode. Continues to start daemons even if an error is encountered.

-pdsh
Enable parallel remote command execution with the PDSH tool. The PDSH tool is required on master hosts and master candidates.

-delay *seconds*
Time interval between chunks. Valid values are 1 - 60. The default value is 8 seconds.

-num_hosts *number*
Number of hosts in one chunk. Valid values are 1 - 512. The default value is 250.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

lsadmin, **badmin**, **lsfrestart**, **lsfshutdown**, `lsf.sudoers`lsf.sudoers

Chapter 79. lsgrun

Runs a task on a group of hosts.

Synopsis

`lsgrun [-i] [-p] [-P] [-S] [-v] -R "res_req" [command [argument ...]]`

`lsgrun [-i] [-p] [-P] [-S] [-v] -f host_file [command [argument ...]]`

`lsgrun [-i] [-p] [-P] [-S] [-v] -m host_name ... [command [argument ...]]`

`lsgrun [-h | -V]`

`lsgrun [-i] [-p] [-P] [-S] [-v] [-R "res_req"] -n num_hosts [command [argument ...]]`

Description

Use the **lsgrun** command for fast global operations such as starting daemons, replicating files to or from local disks, looking for processes that are running on all hosts, checking who is logged in on each host. The hosts can be specified by using a host file, a list of host names or by letting the system select the hosts. If the **LSB_DISABLE_LIMLOCK_EXCL=y** parameter is set to enable preemption of exclusive jobs, for example, you can use the **lsgrun** command to start a task on hosts that are running exclusive jobs.

The **lsgrun** command has the following default behavior:

- The **lsgrun** command is not interactive.
- The specified task runs sequentially on hosts with full pseudoterminal (`tty`) support.
- The **lsgrun** command does not create a pseudo-terminal.
- LSF uses as many processors as available to run the specified task.
- The resource requirement for host selection is `r15s:pg`.
- The prompt `Command>` is displayed to allow users to type in a command as the task. The command prompt is terminated by a `CTRL-D` or `EOF`. The command then runs on the specified hosts.

The `-f host_file`, `-m host_name`, or `-n num_processors` options are required. These options are mutually exclusive.

Options

-i

Interactive operation mode. You are asked whether the task is to run on all hosts. If you answer `y`, the task is started on all specified hosts; otherwise, you are asked to specify hosts interactively.

-P

Creates a pseudo-terminal on UNIX and Linux hosts to run programs that require a pseudo-terminal, for example, the **vi** command.

The `-P` option is not supported on Windows.

-p

Parallel run mode. Runs the task on all hosts simultaneously and without pseudoterminal `tty` support.

If the `-p` option is specified with the `-P` option, the `-P` option is ignored.

Use this option for fast start-up of tasks. However, any output from remote tasks arrives at the terminal in arbitrary order, depending on task execution speeds on individual hosts.

-S

Creates a pseudo-terminal with shell mode support on UNIX hosts.

Shell mode support is required for running interactive shells or applications that redefine the **CTRL-C** and **CTRL-Z keys**, such as **jove**.

The -S option is not supported on Windows.

-v
Verbose mode. Displays the name of the host or hosts running the task.

-f *host_file*
Runs the task on all hosts that are listed in the file that is specified by the *host_file* argument.

Specify a file that contains a list of host names. Host names must be separated by white space characters (for example, SPACE, TAB, and NEWLINE).

This option is exclusive of options -n, -R, and -m.

-m *host_name ...*

Either -f *host_file*, -m *host_name* or -n *num_processors* is required.

Runs the task on all specified hosts.

Specify hosts on which to run the task. If multiple host names are specified, the host names must be enclosed by quotation marks (" or ') and separated by white space.

This option is exclusive of options -n, -R, and -f.

-n *num_hosts*
Runs the task in a cluster with the required number of available hosts.

One host can be used to start several tasks if the host is multiprocessor. This option can be used together with option -R to select hosts.

This option is exclusive of options -m and -f.

-R "*res_req*"
Runs the task on hosts with the required resource requirements.

Specify the resource requirement expression for host selection. Unless a type == value expression is in *res_req* to specify otherwise, the resource requirement is used to choose from all hosts with the same host type as the local host.

The -R option can be used together with option -n to choose a specified number of processors to run the task.

Exclusive resources need to be explicitly specified within the resource requirement string. For example, a resource that is called bigmem is defined in the lsf.shared file and defined it as an exclusive resource for hostE in lsf.cluster.mycluster. Use the following command to submit a task to run on hostE:

```
lsgrun -R "bigmem" myjob
```

Or

```
lsgrun -R "defined(bigmem)" myjob
```

If the -m option is specified with a single host name, the -R option is ignored.

***command* [*argument ...*]**
Specify the command to run. The command must be the last argument on the command line.

-h
Prints command usage to stderr and exits.

-V
Prints LSF release version to stderr and exits.

Diagnostics

Exit status is 0 if all tasks run correctly.

If tasks do not run correct, the exit status is the first nonzero status that is returned by a remote task. The **lsgrun** command runs the task on all hosts even if some have nonzero exit status.

Exit status is -10 if a problem is detected in LSF.

See also

lsrun, **lsplace**

Chapter 80. lshosts

Displays hosts and their static resource information.

Synopsis

`lshosts` [`-a`] [`-cname`] [`-gpu`] [`-w` | `-l` | `-o` "*field_name*[`:`[`-`][*output_width*][`:`*unit*]] ...
[`delimiter='`*character*`']`" [`-json`]] [`-R` "*res_req*"] [`-T`] [*host_name* | *cluster_name*] ...

`lshosts -s` [*resource_name* ...] [`-a`] [`-cname`]

`lshosts` [`-h` | `-V`]

Description

By default, returns the following information: host name, host type, host model, CPU factor, number of CPUs, total memory, total swap space, whether the host is a server host, and static resources. Exclusive resources are prefixed with an exclamation mark (`!`). Displays information about all hosts in the cluster.

In the IBM Spectrum LSF multicluster capability job forwarding model, the default behavior is to return the following information: host name, host type, host model, CPU factor, number of CPUs, total memory, total swap space, whether the host is a server host, and static resources. Displays information about all hosts in the local cluster and for all hosts in equivalent remote clusters that the local cluster sees.

In the IBM Spectrum LSF multicluster capability resource leasing model, returns information about hosts in the local cluster.

The `-s` option displays information about the static resources (shared or host-based) and their associated hosts.

Options

-a
Dynamic cluster only. Shows information about all hosts, including Dynamic cluster virtual machine hosts configured with the jobvm resource. Default output includes only standard LSF hosts and Dynamic cluster hosts configured with the dchost resource.

-cname
In LSF Advanced Edition, includes the cluster name for execution cluster hosts and host groups in output.

-gpu
Displays GPU topology information for the cluster.

-json

Displays the customized output in JSON format.

When specified, **`lshosts -o`** displays the customized output in the JSON format.

This option applies only to output for the **`lshosts -o`** command for customized output. This option has no effect when used with **`lshosts`** without the `-o` option and the **`LSF_LSHOSTS_FORMAT`** environment variable and parameter are not defined.

-l
Displays host information in a long multi-line format. In addition to the default fields, displays additional information, including maximum `/tmp` space, the number of local disks, the execution priority for remote jobs, load thresholds, and run windows.

-w
Displays host information in wide format. Fields are displayed without truncation.

-o

Sets the customized output format.

- Specify which **lshosts** fields (or aliases instead of the full field names), in which order, and with what width to display.
- Specify only the **lshosts** field name or alias to set its output to unlimited width and left justification.
- Specify the colon (:) without a width to set the output width to the recommended width for that field.
- Specify the colon (:) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **lshosts** truncates the ending characters.
- Specify a hyphen (-) to set right justification when **lshosts** displays the output for the specific field. If not specified, the default is to set left justification when **lshosts** displays the output for a field.
- Specify the unit colon (:) with a unit to set the unit for the output of the specific field:
  - Specify S to use a built-in conversion for space or capacity, such as memory or disk space. Values are automatically scaled for M (MB), G (GB), and T (TB), where the default unit is M (MB).

    For example, when displaying the **mem** field with a specified width of 3,

    - For a value of 30, running the **lshosts -o "maxmem:3:S"** command shows 30.0M.
    - For a value of 4096, running the **lshosts -o "maxswp:3:S"** command shows 4.0G.
    - For a value of 5000000, running the **lshosts -o "maxtmp:3:S"** command shows 4.8T.
  - Specify D to use a built-in conversion for duration or time, such as memory or disk space. Values are automatically scaled for s (seconds), m (minutes), h (hours), and d (days), where the default unit is s (seconds). The automatically scaled value is rounded up after the first decimal point.

    For example, when displaying the external **mytime** resource field with a specified width of 5,

    - For a value of 30, running the **lshosts -o "mytime:5:D"** command shows 30.0s.
    - For a value of 8000, running the **lshosts -o "mytime:5:D"** command shows 2.2h.
    - For a value of 5000000, running the **lshosts -o "mytime:5:D"** command shows 57.8d.
  - Specify any other string of 1 - 3 characters and the characters are used as is in the field value. The first character must be a letter (upper or lower case). The second and third characters must be an alphanumeric character.

    For example, when displaying the external gpu_temp resource with a width of 3, running the **lshosts -o "gpu_temp:3:C"** command for a value of 30 shows 30C
- Use delimiter= to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the output for certain **lshosts** options:

- **LSF_LSHOSTS_FORMAT** and **lshosts -o** both apply to output for the **lshosts** command with no options, and for **lshosts** options with output that filter information, including the following options: -a, -cname.
- **LSF_LSHOSTS_FORMAT** and **lshosts -o** do not apply to output for other **lshosts** options that use a modified format, including the following options: -l, -w.

The **lshosts -o** option overrides the **LSF_LSHOSTS_FORMAT** environment variable, which overrides the **LSF_LSHOSTS_FORMAT** setting in lsf.conf.

The following are the field names used to specify the **lshosts** fields to display, recommended width, aliases you can use instead of field names, and units of measurement for the displayed field:

Table 6. Output fields for lshosts

Field name	Width	Aliases	Unit
HOST_NAME	20	hname	

Table 6. Output fields for lshosts (continued)			
Field name	**Width**	**Aliases**	**Unit**
type	10		
model	10		
cpuf	10		
max	10		
ncpus	8		
maxmem	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)
maxswp	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)
server	10		
RESOURCES	20	res	
ndisks	8		
maxtmp	10		**LSF_UNIT_FOR_LIMITS** in `lsf.conf` (KB by default)
rexpri	10		
nprocs	8		
ncores	8		
nthreads	10		
RUN_WINDOWS	20	runwin	

Field names and aliases are not case-sensitive. Valid values for the output width are any positive integer 1 - 4096.

For example,

```
lshosts -o "HOST_NAME type: RESOURCES:- RUN_WINDOWS:-16 delimiter='^'"
```

This command displays the following fields:

- HOST_NAME with unlimited width and left-aligned.
- type with a maximum width of 10 characters (which is the recommended width) and left-aligned.
- RESOURCES with a maximum width of 20 characters (which is the recommended width) and right-aligned.
- RUN_WINDOWS with a maximum width of 16 characters and right-aligned.
- The ^ character is displayed between different headers and fields.

-R "*res_req*"
Displays only information about the hosts that satisfy the resource requirement expression. LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

In the IBM Spectrum LSF multicluster capability, displays only information about the hosts in the local cluster that satisfy the resource requirement expression.

host_name ... | cluster_name ...
Displays only information about the specified hosts. Do not use quotation marks when you specify multiple hosts.

For the IBM Spectrum LSF multicluster capability, displays information about hosts in the specified clusters. The names of the hosts that belong to the cluster are displayed instead of the name of the cluster. Do not use quotation marks when you specify multiple clusters.

-s [*resource_name* ...]
Displays information about the specified resources. The resources must be static resources (shared or host-based). If no resource is specified, then displays information about all resources. Returns the following information: the resource names, the values of the resources, and the resource locations.

-h
Prints command usage to `stderr` and exits.

-T
Displays host topology information for each host or cluster.

-V
Prints the LSF release version to `stderr` and exits.

Host-based default output

HOST_NAME
The name of the host. This display field is truncated.

type
The host type. This display field is truncated.

With the IBM Spectrum LSF multicluster capability, if the host type of a host in the remote cluster is not defined in the local cluster, the keyword `unknown` is displayed.

model
The host model. This display field is truncated.

With the IBM Spectrum LSF multicluster capability, if the host model of a host in the remote cluster is not defined in the local cluster, the keyword `unknown` is displayed.

cpuf
The relative CPU performance factor. The CPU factor is used to scale the CPU load value so that differences in CPU speeds are considered. The faster the CPU, the larger the CPU factor.

The default CPU factor of a host with an `unknown` host type is `1.0`.

ncpus
The number of processors on this host.

If the **LSF_ENABLE_DUALCORE=Y** parameter is specified in the `lsf.conf` file for multi-core CPU hosts, displays the number of cores instead of physical CPUs.

If EGO is enabled in the LSF cluster and the **EGO_DEFINE_NCPUS** parameter is specified in the `lsf.conf` or `ego.conf` file, the appropriate value for `ncpus` is displayed, depending on the value of the **EGO_DEFINE_NCPUS** parameter:

EGO_DEFINE_NCPUS=procs
`ncpus`=number of processors.

EGO_DEFINE_NCPUS=cores
`ncpus`=*number of processors* × *number of cores per processor*.

EGO_DEFINE_NCPUS=threads
`ncpus`=*number of processors* × *number of cores per processor* × *number of threads per core*.

Note: The `EGO_DEFINE_NCPUS=cores` parameter is the same as setting the **LSF_ENABLE_DUALCORE=Y** parameter.

nprocs
The number of physical processors per CPU configured on a host.

ncores
The number of cores per processor that is configured on a host.

nthreads
The number of threads per core that is configured on a host.

maxmem
The maximum amount of physical memory available for user processes.

By default, the amount is displayed in KB. The amount can appear in MB depending on the actual system memory. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for the limit (GB, TB, PB, or EB).

maxswp
The total available swap space.

By default, the amount is displayed in KB. The amount can appear in MB depending on the actual system swap space. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for the limit (GB, TB, PB, or EB).

For the Solaris operating system, the swap space is virtual, a layer between anonymous memory pages and the physical storage (or disk-backed swap space). Virtual swap space on Solaris systems is equal to the sum of all its physical (disk-backed) swap space plus a portion of the currently available physical memory, which might be a dynamic value.

server
Indicates whether the host is a server or client host. `Yes` is displayed for LSF servers. `No` is displayed for LSF clients. `Dyn` is displayed for dynamic hosts.

RESOURCES
The Boolean resources that are defined for this host, denoted by resource names, and the values of external numeric and string static resources. External static resources are configured in the `lsf.cluster` and `lsf.shared` files.

Host-based -l option output

ndisks
The number of local disk drives directly attached to the host.

maxtmp
The maximum `/tmp` space in MB configured on a host.

rexpri
UNIX only. The execution priority of remote jobs that are run by the RES. rexpri is a number between -20 and 20, with -20 representing the highest priority and 20 the lowest. The default rexpri is 0, which corresponds to the default scheduling priority of 0 on BSD-based UNIX systems and 20 on System V-based systems.

nprocs
The number of physical processors per CPU configured on a host.

ncores
The number of cores per processor that is configured on a host.

nthreads
The number of threads per core that is configured on a host.

RUN_WINDOWS
The time windows during which LIM considers the host as available to run remote jobs. These run windows have the same function for LSF hosts as dispatch windows have for LSF hosts.

LOAD_THRESHOLDS
The thresholds for scheduling interactive jobs. If a load index exceeds the load threshold (or falls below the load threshold, for decreasing load indices), the host status is changed to `busy`. If the threshold is displayed as a dash `-`, the value of that load index does not affect the host status.

HARDWARE TOPOLOGY
NUMA and socket information for the host.

AVAILABLE CPU FREQUENCY
Shows the available CPU frequencies for the host. Used for energy aware scheduling.

CURRENT CPU FREQUENCY (GHz)
Shows the current CPU frequencies that are selected for the host and the number of CPUs on the host. Used for energy aware scheduling.

Resource-based output -s option

Displays the static resources (shared or host-based). Each line gives the value and the associated hosts for the static resource. Static shared resources are configured in the `lsf.shared` and `lsf.cluster` files.

The following fields are displayed:

RESOURCE
The name of the resource.

VALUE
The value of the static resource.

LOCATION
The hosts that are associated with the static resource.

Topology-based output -T option

Displays host topology information for each host or cluster. Topology is displayed by *processor unit* level: NUMA node, if present, socket, core, and thread. A *socket* is a collection of cores with a direct pipe to memory. Each socket contains 1 or more cores. The topology display does not necessarily refer to a physical socket, but rather to the memory architecture of the machine. A *core* is a single entity capable of performing computations. On hosts with hyperthreading enabled, a core can contain one or more threads.

The following fields are displayed:

Host[*memory*] *host_name*
Maximum memory available on the host followed by the host name. If memory availability cannot be determined, a dash (-) is displayed for the host.

For hosts that do not support affinity scheduling, a dash (-) is displayed for host memory and no host topology is displayed.

NUMA[*numa_node*: *max_mem*]
Maximum NUMA node memory. It is possible for requested memory for the NUMA node to be greater than the maximum available memory displayed.

If no NUMA nodes are present, then the NUMA layer in the output is not shown. Other relevant items such as host, socket, core, and thread are still shown.

If the host is not available, only the host name is displayed. A dash (-) is shown where available host memory would normally be displayed.

The **`lshosts -T`** output differs from the **`bhosts -aff`** output:

- Socket and core IDs are not displayed for each NUMA node.
- The requested memory of a NUMA node is not displayed
- **`lshosts -T`** displays all enabled CPUs on a host, not just the CPUs defined in the CPU list in `lsb.hosts`

In the following example, full topology (NUMA, socket, and core) information is shown for `hostA`. Hosts `hostB` and `hostC` are either not NUMA hosts or they are not available:

```
lshosts -T
Host[15.7G] hostA
    NUMA[0: 15.7G]
        Socket
            core(0)
        Socket
            core(1)
        Socket
            core(2)
```

```
        Socket
            core(3)
        Socket
            core(4)
        Socket
            core(5)
        Socket
            core(6)
        Socket
            core(7)

Host[-] hostB

Host[-] hostC
```

When LSF cannot detect processor unit topology, the **lshosts -T** command displays processor units to the closest level.

```
lshosts -T
     Host[1009M] hostA
            Socket (0 1)
```

hostA has two processor units: 0 and 1. LSF cannot detect core information, so the processor unit is attached to the socket level.

Hardware topology information is not shown for client hosts and hosts in a mixed cluster or IBM Spectrum LSF multicluster capability environment that is running a version of LSF that is earlier than Version 9.1.

Files

Reads `lsf.cluster.`*`cluster_name`*.

See also

ls_info, **ls_policy**, **ls_gethostinfo**, `lsf.shared`

Chapter 81. lsid

Displays the LSF version number, the cluster name, and the master host name.

Synopsis

```
lsid [-h | -V]
```

Description

The master host is dynamically selected from all hosts in the cluster.

In LSF multicluster capability, the master host is dynamically selected from all hosts in the local cluster.

Options

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Files

The host names are defined in the `lsf.cluster.`*`cluster_name`* file and cluster names are defined in the `lsf.shared` file.

See also

`ls_getclustername`, `ls_getmastername`, **`lsinfo`**

Chapter 82. lsinfo

Displays LSF configuration information.

Synopsis

`lsinfo` [`-l` | `-w`] [`-m` | `-M`] [`-r`] [`-t`] [*resource_name* ...]

`lsinfo` [`-h` | `-V`]

Description

By default, displays all LSF configuration information including resource names and their meanings, host types and models, and associated CPU factors known to the system.

By default, displays information about all resources. Resource information includes resource name, resource type, description, and the default sort order for the resource.

You can use resource names in task placement requests.

Use this command with options to selectively view configured resources, host types, and host models.

Options

-l
Displays resource information in a long multi-line format. Additional parameters are displayed including whether a resource is built-in or configured, and whether the resource value changes dynamically or is static. If the resource value changes dynamically then the interval indicates how often it is evaluated.

-M
Displays information about all host models in the file `lsf.shared`.

-m
Displays only information about host models that exist in the cluster.

-r
Displays only information about configured resources.

-t
Displays only information about configured host types.

-w
Wide format. Displays the information in a wide format.

resource_name ...
Displays only information about the specified resources.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Output for -l option

The `-l` option displays all information available about load indices.

TYPE
Indicates whether the resource is numeric, string, or Boolean.

ORDER

Inc
If the numeric value of the load index increases as the load it measures increases, such as CPU utilization (`ut`).

Dec
If the numeric value decreases as the load increases.

N/A
If the resource is not numeric.

INTERVAL
The number of seconds between updates of that index. Load indices are updated every INTERVAL seconds. A value of 0 means the value never changes.

BUILTIN
If BUILTIN is Yes, the resource name is defined internally by LIM. If BUILTIN is No, the resource name is site-specific defined externally by the LSF administrator.

DYNAMIC
If DYNAMIC is Yes the resource is a load index that changes over time. If DYNAMIC is No the resource represents information that is fixed such as the total swap space on a host. Resources are Static or Boolean.

RELEASE
Applies to numeric shared resources only. Indicates whether LSF releases the resource when a job using the resource is suspended. When a job using a shared resource is suspended, the resource is held or released by the job depending on the configuration of the RELEASE parameter in the lsf.shared file.

No indicates the resource is held. Yes indicates the resource is released.

CONSUMABLE
If CONSUMABLE is Yes, the resource is a static or dynamic numeric resource that is specified as consumable in the Resource section of the lsf.shared file.

See also

lshosts, **lsload**, lsf.sharedlsf.shared, **ls_info**, **ls_policy**

Chapter 83. lsload

Displays load information for hosts.

Synopsis

`lsload` [`-a`] [`-cname`] [`-gpu`] [`-gpuload`] [`-l` | `-w` | `-o` "*field_name*[`:`[`-`][*output_width*]] ... [`delimiter='`*character*`']"` [`-json`]] [`-N` | `-E`] [`-I` *load_index*[`:`*load_index*] ...] [`-n` *num_hosts*] [`-R` *res_req*] [*host_name* ... | *cluster_name* ...]

`lsload -s` [*resource_name* ...] [`-cname`]

`lsload` [`-h` | `-V`]

Description

Load information can be displayed on a per-host basis, or on a per-resource basis.

By default, displays load information for all hosts in the local cluster.

With the IBM Spectrum LSF multicluster capability enabled, the **lsload** command also displays load information for all hosts in equivalent clusters.

By default, displays raw load indices.

By default, load information for resources is displayed according to CPU and paging load.

Options

-a
Dynamic Cluster only. Shows information about all hosts, including Dynamic Cluster virtual machine hosts configured with the jobvm resource. Default output includes only standard LSF hosts and Dynamic Cluster hosts configured with the `dchost` resource.

-cname
In LSF Advanced Edition, includes the cluster name for execution cluster hosts and host groups in output.

-gpu
Displays host-based GPU information.

-gpuload
Displays GPU-based information.

-json
Displays the customized output in JSON format.

When specified with the `-o` option, displays the customized output in the JSON format.

This option should be used with the `-o` option. If this option is not used with the `-o` option, the **LSF_LSLOAD_FORMAT** parameter in the `lsf.conf` file or the **LSF_LSLOAD_FORMAT** environment variable must be defined.

-l
Long format. Displays load information without truncation along with extra fields for I/O and external load indices.

This option overrides the index names that are specified with the `-I` option.

-N
Displays normalized CPU run queue length load indices.

-E
Displays effective CPU run queue length load indices. Options `-N` and `-E` are mutually exclusive.

-w
Displays load information in wide format. Fields are displayed without truncation.

-I *load_index*[:*load_index*] ...
Displays only the specified load indices. Separate multiple index names with colons (for example, `r1m:pg:ut`).

Specify any built-in load index. Specify external load indices only for host-based resources that are numeric and dynamic (you cannot specify external load indices for shared, string, or Boolean resources).

-n *num_hosts*
Displays only load information for the requested number of hosts. Information for up to num_hosts hosts that best satisfy the resource requirements is displayed.

-o

Sets the customized output format.

- Specify which **lsload** fields, in which order, and with what width to display.
- Specify the asterisk wildcard character (*****) in the field name to specify multiple external resource names. You can only specify one asterisk, but this asterisk can be at any position in the field name.

  For example, running **lsload -o "gpu_mode*"** shows fields such as `gpu_mode0`, `gpu_mode1`, `gpu_mode2`, `gpu_model0`, `gpu_model1`, and `gpu_model2`.
- Specify only the **lsload** field name to set its output to unlimited width and left justification.
- Specify the width colon (`:`) without a width to set the output width to the recommended width for that field.
- Specify the width colon (`:`) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **lsload** truncates the ending characters.
- Specify a hyphen (`-`) to set right justification when **lsload** displays the output for the specific field. If not specified, the default is to set left justification when **lsload** displays the output for a field.
- Specify the unit colon (`:`) with a unit to set the unit for the output of the specific field:
  - Specify `S` to use a built-in conversion for space or capacity, such as memory or disk space. Values are automatically scaled for `M` (MB), `G` (GB), and `T` (TB), where the default unit is `M` (MB).

    For example, when displaying the **mem** field with a specified width of 3,

    - For a value of 30, running the **lsload -o "mem:3:S"** command shows `30.0M`.
    - For a value of 4096, running the **lsload -o "mem:3:S"** command shows `4.0G`.
    - For a value of 5000000, running the **lsload -o "mem:3:S"** command shows `4.8T`.
  - Specify `D` to use a built-in conversion for duration or time, such as memory or disk space. Values are automatically scaled for s (seconds), m (minutes), h (hours), and d (days), where the default unit is s (seconds). The automatically scaled value is rounded up after the first decimal point.

    For example, when displaying the external **mytime** resource field with a specified width of 5,

    - For a value of 30, running the **lsload -o "mytime:5:D"** command shows `30.0s`.
    - For a value of 8000, running the **lsload -o "mytime:5:D"** command shows `2.2h`.
    - For a value of 5000000, running the **lsload -o "mytime:5:D"** command shows `57.8d`.
  - Specify any other string of 1 - 3 characters and the characters are used as is in the field value. The first character must be a letter (upper or lower case). The second and third characters must be an alphanumeric character.

    For example, when displaying the external `gpu_temp` resource with a width of 3, running the **lsload -o "gpu_temp:3:C"** command for a value of 30 shows `30C`
- Use `delimiter=` to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the output for certain **lsload** options:

- **LSF_LSLOAD_FORMAT** and **lsload -o** both apply to output for the **lsload** command with no options, and for **lsload** options with short form output that filter information, including the following options: -a, -E, -cname, -N, -n, -R.
- **LSF_LSLOAD_FORMAT** and **lsload -o** do not apply to output for **lsload** options that use a modified format, including the following options: -I, -l, -w, -s.

The **lsload -o** option overrides the **LSF_LSLOAD_FORMAT** environment variable, which overrides the **LSF_LSLOAD_FORMAT** setting in lsf.conf.

By default, the **lsload** command displays the built-in resource indices. You can also specify the names of external resources. The following are the field names for the built-in resource indices that are used to specify the **lsload** fields to display, recommended width, and units of measurement for the displayed field:

Table 7. Output fields for lsload

Field name	Width	Unit
HOST_NAME	20	
status	15	
r15s	6	
r1m	6	
r15m	6	
ut	6	
pg	6	
ls	6	
it	6	
io	6	
tmp	10	**LSF_UNIT_FOR_LIMITS** in lsf.conf (KB by default)
swp	10	**LSF_UNIT_FOR_LIMITS** in lsf.conf (KB by default)
mem	10	**LSF_UNIT_FOR_LIMITS** in lsf.conf (KB by default)
gpu_status* For example, gpu_status0 and gpu_status1 if there are two GPUs.	10	
gpu_error* For example, gpu_error0 and gpu_error1 if there are two GPUs.	20	

Field names are case-sensitive. Valid values for the output width are any positive integer 1 - 4096.

For example,

```
lsload -o "HOST_NAME status: r15s:- r1m:7 r15m:-8 tmp:S swp::S mem:9:S delimiter='^'"
```

This command displays the following fields:

- HOST_NAME with unlimited width and left-aligned.

- status with a maximum width of 15 characters (which is the recommended width) and left-aligned.
- r15s with a maximum width of 6 characters (which is the recommended width) and right-aligned.
- r1m with a maximum width of 7 characters and left-aligned.
- r15m with a maximum width of 8 characters and right-aligned.
- tmp with unlimited width, left-aligned, and automatically scaled for space or capacity (MB, GB, and TB).
- swp with a maximum width of 10 characters (which is the recommended width), left-aligned, and automatically scaled for space or capacity (MB, GB, and TB)
- mem with a maximum width of 9 characters, left-aligned, and automatically scaled for space or capacity (MB, GB, and TB)
- The ^ character is displayed between different headers and fields.

-R *res_req*
Displays only load information for hosts that satisfy the specified resource requirements.

Load information for the hosts is sorted according to load on the specified resources.

If *res_req* contains special resource names, only load information for hosts that provide these resources is displayed (run **lshosts** to find out what resources are available on each host).

If one or more host names are specified, only load information about the hosts that satisfy the resource requirements is displayed.

With the IBM Spectrum LSF multicluster capability, when a cluster name is specified, displays load information of hosts in the specified cluster that satisfy the resource requirements.

***host_name* ... | *cluster_name* ...**
Displays only load information for the specified hosts.

With the IBM Spectrum LSF multicluster capability, displays only load information for hosts in the specified clusters.

-s [*resource_name* ...]
Displays information about all dynamic resources that are configured in the cluster, or about the specified resources only. Specify dynamic resources (shared or host-based).

-h
Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Default host-based output

Built-in load indices include `r15s`, `r1m`, `r15m`, `ut`, `pg`, `io`, `ls`, `it`, `swp`, `mem`, and `tmp`. External load indices are configured in the file `lsf.cluster.`*cluster_name*. The selection and order sections of resource requirements control for which hosts are displayed and how the information is ordered.

The display includes the following fields:

HOST_NAME
Standard host name that is used by LSF, typically an internet domain name with two components.

status
Status of the host. A minus sign (-) can precede the status, indicating that RES is not running on the host.

The following statuses are displayed:

ok
The host is in normal state and can accept remote jobs. The ok status indicates that the Load Information Manager (LIM) is unlocked and that both LIM and the Remote Execution Server (RES) are running.

-ok
The LIM on the host is running but RES is unreachable.

busy
The host is overloaded because some load indices exceed configured thresholds. Load index values that caused the host to be busy are preceded by an asterisk (*).

lockW
The host is locked by its run window. Run windows for a host are specified in the `lsf.conf` configuration file, and can be displayed by the **lshosts** command. A locked host does not accept LSF jobs from other hosts.

lockU
The host is locked by the LSF administrator or root.

unavail
The host is down or the LIM on the host is not running.

r15s
The 15 second exponentially averaged CPU run queue length.

r1m
The 1 minute exponentially averaged CPU run queue length.

r15m
The 15 minute exponentially averaged CPU run queue length.

ut
The CPU utilization exponentially averaged over the last minute, 0 - 1.

io
By default, `io` is not shown.

If the `-l` option is specified, shows the disk I/O rate exponentially averaged over the last minute, in KB per second.

pg
The memory paging rate exponentially averaged over the last minute, in pages per second.

ls
The number of current login users.

it
On UNIX, the idle time of the host (keyboard is not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time that a screen saver is active on a particular host.

tmp
The amount of free space in `/tmp`, in MB.

swp
The amount of available swap space.

By default, the amount is displayed in KB. The amount can appear in MB depending on the actual system swap space. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for the limit (GB, TB, PB, or EB).

mem
The amount of available RAM.

By default, the amount is displayed in KB. The amount can appear in MB depending on the actual system memory. Use the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file to specify a larger unit for the limit (GB, TB, PB, or EB).

external_index
By default, external load indices are not shown.

If the `-l` option is specified, shows indices for all dynamic custom resources available on the host, including shared, string, and Boolean resources.

If the `-I load_index` option is specified, shows only indices for specified non-shared (host-based) dynamic numeric custom resources.

Resource-based output with lsload -s

Displays information about shared and host-based dynamic resources. Each line gives the value and the associated hosts for an instance of the resource.

The displayed information consists of the following fields:

RESOURCE
Name of the resource.

VALUE
Value for an instance of the resource.

LOCATION
Hosts associated with the instance of the resource.

Network resource information with lsload -l

If the **LSF_PE_NETWORK_NUM** parameter is set to a value greater than zero in the `lsf.conf` file, LSF collects network information for scheduling IBM Parallel Environment (PE) jobs. Two string resources are created for PE jobs:

pe_network
A host-based string resource that contains the network ID and the number of network windows available on the network.

pnsd
Set to Y if the PE network resource daemon **pnsd** responds successfully, or N if the daemon does not respond. PE jobs can run only on hosts with **pnsd** installed.

The **lsload -l** command displays the value of these two resources and shows network information for PE jobs. For example, the following **lsload** command displays network information for hostA and hostB, both of which have two networks available. Each network has 256 windows, and **pnsd** is responsive on both hosts. In this case, the LSF_PE_NETWORK_NUM=2 parameter is set in the `lsf.conf` file:

```
lsload -l
HOST_NAME   status  r15s   r1m  r15m   ut    pg    io  ls    it   tmp   swp   mem   pnsd
pe_network
hostA              ok   1.0   0.1   0.2  10%   0.0     4  12     1   33G 4041M 2208M  Y
ID= 1111111,win=256;ID= 2222222,win=256
hostB              ok   1.0   0.1   0.2  10%   0.0     4  12     1   33G 4041M 2208M  Y
ID= 1111111,win=256;ID= 2222222,win=256
```

Examples

The following command displays the load of ALPHA hosts with at least 20 MB of swap space, and a 1-minute run queue length less than 0.5.

```
lsload -R "select[r1m<=0.5 && swp>=20 && type==ALPHA]"
```

The following command specifies the same resource requirements in restricted format:

```
lsload -R r1m=0.5:swp=20:type=ALPHA
```

The following command displays the load of the hosts whose swap space utilization is less than 75%. The resulting hosts are ordered by paging rate.

```
lsload -R "select[(1-swp/maxswp)<0.75] order[pg]"
```

The following command displays the 1-minute CPU raw run queue length, the CPU utilization, the disk I/O, and paging rates for all hosts in the cluster.

```
lsload -I r1m:ut:io:pg
```

The following command displays the load of all hosts, ordered by `r15s:pg`. The CPU run queue lengths are the effective run queue lengths.

```
lsload -E
```

Diagnostics

Exit status is -10 for LSF problems or an invalid resource names.

Exit status is -1 if an invalid option is specified.

Normal exit status for the **lsload** command is 0.

See also

lim, `lsf.cluster`, **lsplace**, **lshosts**, **lsinfo**, **lslockhost**, **ls_load**

Chapter 84. lsloadadj

Adjusts load indices on hosts.

Synopsis

`lsloadadj` [`-R` *res_req*] [*host_name*[`:`*num_task*] ...]

`lsloadadj` [`-h` | `-V`]

Description

Adjusts load indices on hosts. This is useful if a task placement decision is made outside LIM by another application.

By default, assumes tasks are CPU-intensive and memory-intensive. This means the CPU and memory load indices are adjusted to a higher number than other load indices.

By default, adjusts load indices on the local host, the host from which the command was submitted.

By default, starts 1 task.

Upon receiving a load adjustment request, LIM temporarily increases the load on hosts according to resource requirements. This helps LIM avoid sending too many jobs to the same host in quick succession. The adjusted load decays over time before the real load produced by the dispatched task is reflected in LIM's load information.

The **`lsloadadj`** command adjusts all indices except for `ls` (login sessions), `it` (idle time), `r15m` (15 minute run queue length) and external load indices. Other load indices can only be adjusted beyond specific maximum values.

- `tmp` is -0.5
- `swp` is -1.5
- `mem` is -1.0
- `r1m` is 0.4
- `ut` is 15%
- `r15s` is 0.1
- `pg` is 0.3

Options

-R *res_req*
Specify resource requirements for tasks. Only the resource usage (`rusage`) section of the resource requirement string is considered. This is used by LIM to determine by how much individual load indices are to be adjusted.

For example, if a task is swap-space-intensive, load adjustment on the `swp` load index is higher; other indices are increased only slightly.

***host_name*[:*num_task*] ...**
Specify a list of hosts for which load is to be adjusted. num_task indicates the number of tasks to be started on the host.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Examples

```
lsloadadj -R "rusage[swp=20:mem=10]"
```

Adjusts the load indices swp and mem on the host from which the command was submitted.

Diagnostics

Returns -1 if an invalid parameter is specified; otherwise returns 0.

See also

lsinfo, **lsplace**, **lsload**, ls_loadadj

Chapter 85. lslogin

Remotely logs in to a lightly loaded host.

Synopsis

`lslogin [-v] [-m "host_name ..." | -m "cluster_name ..."] [-R "res_req"] [rlogin_options]`

`lslogin [-h | -V]`

Description

By default, the **lslogin** command selects the least loaded host, with few users who are logged in, and remotely logs in to that host by using the UNIX **rlogin** command.

In a IBM Spectrum LSF multicluster capability environment, the default is to select the least loaded host in the local cluster.

As an alternative to the **rlogin** command, you can use an SSH connection by enabling the **LSF_LSLOGIN_SSH** parameter in the `lsf.conf` file.

Options

-v
Displays the name of the host that the **lslogin** command remotely logs you in to.

-m "*host_name ...*" | -m "*cluster_name ...*"
Remotely logs in to the specified host.

With the IBM Spectrum LSF multicluster capability job forwarding, when a cluster name is specified, remotely logs in to the least loaded host in the specified cluster. The remote cluster must be configured to accept interactive jobs from the local cluster in the `lsf.cluster` file.

-R "*res_req*"
Remotely logs in to a host that meets the specified resource requirement. The resource requirement expression restricts the set of candidate hosts and determines the host selection policy.

To find out what resources are configured in your system, use the **lsinfo** and **lshosts** commands.

rlogin_options
Specify remote login options that are passed to the **rlogin** command.

If remote execution fails, the **lslogin** command logs in locally only if the local host also satisfies required resources; otherwise, log in fails.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Example

```
lslogin -R "select[it>1 && bsd]"
```

Remotely logs in to a host that is idle for at least 1 minute, runs BSD UNIX, and is lightly loaded both in CPU resources and the number of users who are logged in.

Diagnostics

Because the **lslogin** command passes all unrecognized arguments to the **rlogin** command, incorrect options usually cause the **rlogin** usage message to be displayed rather than the **lslogin** usage message.

See also

ls_placereq, **rlogin**

Chapter 86. lsltasks

Displays or updates a local task list.

Synopsis

`lsltasks` [+ *task_name* ... | – *task_name* ...]

`lsltasks` [`-h` | `-V`]

Description

Displays or updates a user local task list in the `$HOME/.lsftask` file.

When no options are specified, displays tasks that are listed in the system task file `lsf.task` and the user task file `.lsftask`.

If the system task file `lsf.task` conflicts with the user task file `.lsftask`, the user task file overrides the system task file.

Tasks in the local task list are not eligible for remote execution, either because they are trivial tasks or because they need resources on the local host.

Options

+ *task_name*
If a plus sign (+) is specified and the specified task names are not already in the user task file (`.lsftask`), adds the task names to the file with a plus sign (+) preceding them.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a + or nothing, replaces the entry with the specified content; if the entry starts with a minus sign (`-`), deletes the entry from the `.lsftask` file.

- *task_name*
If a minus sign (–) is specified and specified task names are not already in the user `.lsftask` file, adds the task names to the file with a – preceding the task name.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a –, no operation is done; if the entry starts with a +, deletes the entry from the `.lsftask` file.

-h
Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

```
lsltasks + foo
```

Adds the command `foo` to the local task list.

Files

Reads the system task file `lsf.task`, and the user `.lsftask` file.

The system and user task files contain two sections, one for the remote task list, the other for the local task list. The local tasks section starts with `Begin LocalTasks` and ends with `End LocalTasks`. Each line in the section is a task name.

A plus sign (+) or a minus sign (–) can optionally precede each entry. If no + or – is specified, then + is assumed.

See also

lseligible, **lsrtasks**, lsf.task, **ls_eligible**, **ls_task**

Chapter 87. lsmake

Runs LSF **make** tasks in parallel.

Synopsis

`lsmake` [`-m` "*host_name* [*num_cores*] [*host_name* [*num_cores*]] ...]"] [`-a` *seconds*] [`-c` *num_tasks*] [`-E`] [`-G` *debug_level*] [`-T`] [`-u`] [`-V`] [`-x` *num_retries*] [`-y`] [*make_option* ...] [`--no-block-shell-mode`] [*target* ...]

`lsmake` [`-R` *res_req*] [`-j` *max_cores*] [`-a` *seconds*] [`-c` *num_tasks*] [`-E`] [`-G` *debug_level*] [`-T`] [`-u`] [`-V`] [`-x` *num_retries*] [`-y`] [*make_option* ...] [`--no-block-shell-mode`] [*target* ...]

`lsmake` [`-h`]

Description

Runs make tasks in parallel on LSF hosts. Sets the environment variables on the remote hosts when the **lsmake** command starts.

By default, uses the local host, uses only one core, starts only one task in each core, processes submakes sequentially, allows 1-second buffer time to compensate for file system latency, and does not retry if the job fails. The **lsmake** command is a modified version of the GNU **make** command.

Options

-a *seconds*
When commands in a target finish, commands in a dependent target wait the specified time before they start on a different host. This delay allows time for the shared file system to synchronize client and server, and compensates for file system latency. By default, the delay is 1 second. Slower file systems require a longer delay.

If the dependent target's commands start on the same execution host, there is no delay.

If retries are enabled with `-x`, the interval between retries also depends on the delay time.

-c *num_tasks*
Starts the specified number of tasks concurrently on each core. If you specify too many tasks, you might overload a host.

-E
Sets the environment variables for every task sent remotely.

Setting environment variables is necessary when makefiles change or override the environment variables that they inherit at startup.

-G *debug_level*
Enables debugging, specify the debug level.

-j *max_cores*
Uses multiple cores, selecting the best available. Specify the maximum number of cores to use.

Not compatible with `-m "host_name [num_cores] [host_name [num_cores]] ..."` option.

Ignored if you use the **bsub** command to run the **lsmake** command.

-m *"host_name* [*num_cores*] [*host_name* [*num_cores*]] ..."
Uses the specified hosts. To use multiple cores on a host, specify the number of cores after the host name.

Not compatible with `-R` *res_req* and `-j` *max_cores*.

Ignored if you use **bsub** to run **lsmake**.

-R ***res_req***
Uses only hosts that satisfy the specified resource requirements.

When you specify `-R` but not `-j`, uses one core on one host that satisfies the resource requirements.

If the group of hosts that match the `selection` string includes the submission host, the submission host is always selected. The policies that are defined by the `order` string affect only the other hosts.

Not compatible with the `-m "`*`host_name`* `[`*`num_cores`*`] [`*`host_name`* `[`*`num_cores`*`]] ..."` option.

Ignored if you use **bsub** to run **lsmake**.

-T
Enables output tagging to prefix the task ID of the sender to the parallel task output data.

-u
Creates the data file `lsmake.dat` and updates it each second, tracking the number of running tasks over time.

The `lsmake.dat` file is useful if you want to export the data to third-party charting applications.

-V
Verbose mode. Prints the names of the hosts used.

-x ***num_retries***
If the command fails, retries the command the specified number of times. For example, if the number of retries is 1, the command is attempted twice before it exits. Setting retries is useful to compensate for file system latency and minor errors.

The interval between retries increases exponentially with each retry attempt. The time between the initial, failed attempt, and the first retry is equal to 1 second by default, or equal to the buffer time specified by `-a`. For subsequent attempts, the interval between attempts is doubled each time.

-y
Displays summary information after the job is done.

make_option ...
Specifies standard GNU Make options.

Note: The `-j` and `-R` options are not supported as a GNU make options, see the **lsmake** options `-j` *`max_cores`* and `-R` *`res_req`*. See GNU documentation for detailed descriptions of other options.

The current version of the **lsmake** command supports GNU Make version 3.81, which includes the following options:

-b, -m
Ignored for compatibility.

-B, --always-make
Unconditionally make all targets.

-C ***dir*****, --directory=***dir*
Change directory before reading the makefile.

-d
Print all debugging information.

--debug[=*options***]**
Print basic debugging information, or specify what types of information to print (`all`, `basic`, `verbose`, `implicit`, `jobs`, `makefile`).

-e, --environment-overrides
Environment variables override makefiles.

-f ***file*****, --file=***file*****, --makefile=***file*
Specify the makefile.

-h, --help
Print usage and exit.

-i, --ignore-errors
Ignore errors.

-I *dir*, --include-dir=*dir*
Search a directory for included makefiles.

-k, --keep-going
Keep going when some targets cannot be made.

-l [*n*], --load-average[=*n*], --max-load[=*n*]
Obsolete. Load limit.

-L , --check-symlink-times
Target file modification time considers the time stamp of symbolic links also.

-n, --just-print, --dry-run, --recon
Print instead of running.

-o *file*, --old-file=*file*, --assume-old=*file*
Do not remake the old file.

-p, --print-data-base
Print make's internal database.

-q, --question
Question mode, return exit status.

-r, --no-builtin-rules
Disable the built-in implicit rules.

--no-builtin-variables
Disable the built-in variable settings. The **make** command option `-R` is not supported, it conflicts with the **lsmake** command option `-R` *res_req*.

-s, --silent, --quiet
Silent mode, do not echo commands.

-S, --no-keep-going, --stop
Turns off the `-k` option.

-t, --touch
Touch targets (change modification time) instead of remaking them.

-v, --version
Print the version number of make and exit.

-w, --print-directory
Print the current directory.

--no-print-directory
Turn off `-w`, even if it was turned on implicitly.

-W *file*, --what-if=*file*, --new-file=*file*, --assume-new=*file*
Always consider the file to be new. Do not change modification time.

--warn-undefined-variables
Warn when an undefined variable is referenced.

--no-block-shell-mode
Run child shell tasks without blocking mode. Without this parameter, blocking mode is used. Allows the **lsmake** command to build customized Android 4.3 code.

***target* ...**
Specifies targets to make.

Output with the -y option

Total Run Time
Total **lsmake** job run time, in the format *hh*:*mm*:*ss*.

Most Concurrent Tasks
Maximum number of tasks that ran simultaneously. Compare to `Total Slots Allocated` and `Tasks Allocated per Slot` to determine whether parallel execution might be limited by resource availability.

Retries Allowed
Maximum number of retries allowed. Set by the `lsmake -x` option.

Hosts and Number of Slots Allocated
The output is a single line that shows each name and number pair that is separated by spaces, in the format: *host_name number_slots* [*host_name number_slots*] ...

Tasks Allowed per Slot
Maximum number of tasks that are allowed per slot. Set by the `lsmake -c` option.

Total Slots Allocated
Total number of slots allocated. Might be limited by the **lsmake -j** or **lsmake -m** option.

Output with the -u option

The `lsmake.dat` file is a simple text file, consisting of two values separated by a comma. The first value is the time in the format *hh:mm:s*. The second value is the number of running tasks. For example:

```
23:13:39,2
```

The file is updated with a new line of information every second.

Limitations

If a submake in a makefile specifies options that are specific to the **lsmake** command, they are ignored. Only the command-line options are used. The resource requirements of tasks in the remote task list are not considered when **lsmake** dispatches tasks.

See also

lstcsh, **gmake**

Chapter 88. lsmon

Displays load information for LSF hosts and periodically updates the display.

Synopsis

`lsmon` [`-N` | `-E`] [`-n` *num_hosts*] [`-R` *res_req*] [`-I` *index_list*] [`-i` *interval*] [`-L` *file_name*] [*host_name* ...]

`lsmon` [`-h` | `-V`]

Description

The **`lsmon`** command is a full-screen LSF monitoring utility that displays and updates load information for hosts in a cluster.

By default, displays load information for all hosts in the cluster, up to the number of lines that fit on-screen.

By default, displays raw load indices.

By default, load information is sorted according to CPU and paging load.

By default, load information is updated every 10 seconds.

Options

-N
Displays normalized CPU run queue length load indices.

-E
Displays effective CPU run queue length load indices. Options `-N` and `-E` are mutually exclusive.

-n *num_hosts*
Displays only load information for the requested number of hosts. Information for up to num_hosts hosts that best satisfy resource requirements is displayed.

-R *res_req*
Displays only load information for hosts that satisfy the specified resource requirements.

Load information for the hosts is sorted according to load on the specified resources.

If resource requirement *res_req* contains special resource names, only load information for hosts that provide these resources is displayed. Use the **`lshosts`** command to find out what resources are available on each host.

If one or more host names are specified, only load information for the hosts that satisfy the resource requirements is displayed.

-I *index_list*
Displays only load information for the specified load indices. Load index names must be separated by a colon (for example, `r1m:pg:ut`).

If the index list *index_list* is too long to fit in the screen, the output is truncated. For example, if the invoker's screen is 80 characters wide, then up to 10 load indices are displayed.

-i *interval*
Sets how often load information is updated on-screen, in seconds.

-L *file_name*
Saves load information in the specified file while it is displayed on-screen.

If you do not want load information to be displayed on your screen at the same time, use `lsmon -L` *`file_name`* `< /dev/null`. The format of the file is described in the `lim.acct` file reference.

host_name ...
Displays only load information for the specified hosts.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Usage

You can use the following commands while **lsmon** is running:

[^L | i | n | N | E | R | q]

^L
Refreshes the screen.

i
Prompts you to input a new update interval.

n
Prompts you to input a new number of hosts to display.

N
Toggles between displaying raw CPU run queue length load indices and normalized CPU run queue length load indices.

E
Toggles between displaying raw CPU run queue length load indices and effective CPU run queue length load indices.

R
Prompts you to input new resource requirements.

q
Quits **lsmon**.

Output

The following fields are displayed by default.

HOST_NAME
Name of specified hosts for which load information is displayed, or if resource requirements were specified, name of hosts that satisfied the specified resource requirement and for which load information is displayed.

status
Status of the host. A minus sign (-) can precede the status, indicating that the Remote Execution Server (RES) on the host is not running.

The following statuses are supported:

ok
The host is in normal load-sharing state and can accept remote jobs.

busy
The host is overloaded because some load indices exceed configured thresholds. Load index values that caused the host to be busy are preceded by an asterisk (*. Built-in load indices include `r15s`, `r1m`, `r15m`, `ut`, `pg`, `io`, `ls`, `it`, `swp`, `mem`, and `tmp`. External load indices are configured in the file `lsf.cluster.`*`cluster_name`*.

lockW
The host is locked by its run window. Run windows for a host are specified in the `lsf.conf` file and can be displayed by the **lshosts** command. A locked host does not accept load shared jobs from other hosts.

lockU
The host is locked by the LSF administrator or root.

unavail
The host is down or the Load Information Manager (LIM) on the host is not running.

unlicensed
The host does not have a valid LSF license.

r15s
The 15 second exponentially averaged CPU run queue length.

r1m
The 1 minute exponentially averaged CPU run queue length.

r15m
The 15 minute exponentially averaged CPU run queue length.

ut
The CPU utilization exponentially averaged over the last minute, 0 - 1.

pg
The memory paging rate exponentially averaged over the last minute, in pages per second.

ls
The number of current login users.

it
On UNIX, the idle time of the host (the keyboard was not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver is active on a particular host.

tmp
The amount of free space in `/tmp`, in MBytes.

swp
The amount of currently available swap space, in MB.

mem
The amount of currently available memory, in MB.

Diagnostics

Specifying an incorrect resource requirement string with the R option while the **lsmon** command is running causes the **lsmon** command to exit with an error message.

lsmon exits if it does not receive a reply from LIM within the update interval.

See also

lshosts, **lsinfo**, **lsload**, **lslockhost**, `lim.acct`, **ls_load**

Chapter 89. lspasswd

Registers Windows user passwords in LSF. Passwords must be 3 - 23 characters long.

Synopsis

`lspasswd` [`-u` *DOMAIN_NAME**user_name*] [`-p` *password*] [`-t` *host_type*]

`lspasswd` [`-r`] [`-u` *DOMAIN_NAME**user_name*] [`-t` *host_type*]

`lspasswd` [`-c`] [`-u` *DOMAIN_NAME**user_name*] [`-t` *host_type*]

`lspasswd` [`-h` | `-V`]

Description

All users can create and verify passwords; only the LSF administrator and root can delete passwords.

Users must update the password that is maintained by LSF if they change their Windows user account password.

Passwords are Windows user account passwords and are saved in the LSF database. LSF uses the passwords to start jobs on behalf of the user. Passwords are stored in encrypted format and the password database is protected by file access permissions. Passwords remain encrypted as they travel through the network.

From UNIX platforms, the option `-u` *DOMAIN_NAME**user_name* must be entered inside double quotation marks "*DOMAIN_NAME**user_name*" or with a double backslash *DOMAIN_NAME**user_name* to avoid reading backslash ("\") as an escape character.

The `-p` option allows scripts to use the **lspasswd** command. Do not use this option directly on the command line because the password is entered in full view on the command line. Only error messages are displayed with the `-p` option.

Specify the `-t` option (identifying a Windows server host type) only if you are both running the **lspasswd** command from a UNIX host and you defined customized Windows host types other than the defaults. The specified host type can be any existing Windows server host type, it does not have to be the execution host type.

The `-t` option is not needed and is ignored if run from a Windows host.

Options

-c -u *DOMAIN_NAMEuser_name* -t *host_type***
Check that the password that is saved in LSF is valid for the specified user.

-r -u *DOMAIN_NAMEuser_name* -t *host_type***
Remove the user entry from the password database.

-u *DOMAIN_NAMEuser_name* -p *password* -t *host_type***
Specify the user and password for the user whose password you want to register or change.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Chapter 90. lsplace

Displays hosts available to run tasks.

Synopsis

`lsplace` [`-L`] [`-n` *minimum* | `-n 0`] [`-R` *res_req*] [`-w` *maximum* | `-w 0`] [*host_name* ...]

`lsplace` [`-h` | `-V`]

Description

Displays hosts available for the execution of tasks, and temporarily increases the load on these hosts (to avoid sending too many jobs to the same host in quick succession). The inflated load decays slowly over time before the real load that is produced by the dispatched task is reflected in the LIM's load information. Host names might be duplicated for multiprocessor hosts to indicate that multiple tasks can be placed on a single host.

By default, displays only one host name.

By default, uses LSF default resource requirements.

Options

-L
Attempts to place tasks on as few hosts as possible. Use the `-L` option for distributed parallel applications to minimize communication costs between tasks.

-n *minimum* | -n 0
Displays at least the specified number of hosts. Specify 0 to display as many hosts as possible.

Prints `Not enough host(s) currently eligible` and exits with status 1 if the required number of hosts with the required resources cannot be found.

-R *res_req*
Displays only hosts with the specified resource requirements. When LSF_STRICT_RESREQ=Y in `lsf.conf`, LSF rejects resource requirement strings where an `rusage` section contains a non-consumable resource.

-w *maximum* | -w 0
Displays no more than the specified number of hosts. Specify 0 to display as many hosts as possible.

***host_name* ...**
Displays only hosts that are among the specified hosts.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Examples

The **`lsplace`** command is mostly used in backward quotation marks (‘) to pick out a host name that is then passed to other commands. The following example issues a command to display a lightly loaded HPPA-RISC host for your program to run on:

```
lsrun -m ‘lsplace -R hppa‘ myprogram
```

In order for a job to land on a host with an exclusive resource, you need to explicitly specify that resource for the resource requirements. The following example issues a command to display the host with the `bigmem` exclusive resource for your program to run on:

```
lsrun -m ‘lsplace -R "bigmem"‘ myprogram
```

The `-w` (upper bound) and `-n` (lower bound) options can be combined to specify the range of processors to be returned. For example, the following command returns at least 3 and not more than 5 host names.

```
lsplace -n 3 -w 5
```

Diagnostics

The **lsplace** command returns 1 if not enough hosts are available. The exit status is -10 if a problem is detected in LSF. Exit status is -1 for other errors. Normal exit status is 0.

See also

lsinfo, **lsload**, **lsrun**, **ls_placereq**

Chapter 91. lsportcheck

Displays ports that LSF is currently using or the LSF ports that will be used before starting LSF.

Synopsis

```
lsportcheck -l [-m | -s]
lsportcheck [-h]
```

Description

In UNIX hosts, you must run the **lsportcheck** command as root to get complete information on the ports in the operating system.

The **lsportcheck** command displays both TCP and UDP ports.

Options

-l [-m]
Displays TCP and UDP ports on the LSF master host. Run this command option on the LSF master host.

-l -s
Displays TCP and UDP ports on the LSF server host. Run this command option on the LSF server host.

-h
Prints command usage and exits.

Chapter 92. lsrcp

Remotely copies files through LSF.

Synopsis

`lsrcp` [`-a`] *source_file target_file*

`lsrcp` [`-h` | `-V`]

Description

The **lsrcp** command is a load-sharing remote copy program that transfers a single file between hosts in an LSF cluster. The **lsrcp** command uses RES on an LSF host to transfer files. If LSF is not installed on a host or if RES is not running, then **lsrcp** uses **rcp** to copy the file.

If the **LSF_REMOTE_COPY_CMD** parameter is defined in the `lsf.conf` file, the **lsrcp** command uses the specified command and options to copy the file if the RES cannot copy the file.

To use **lsrcp**, you must have read access to the file to be copied.

Both the source and target file must be owned by the user who enters the command.

The **lsrcp** command uses the **rcp** command to copy a source file to a target file owned by another user.

Options

-a
Appends source_file to target_file.

source_file target_file
Specify an existing file on a local or remote host that you want to copy, and a file to which you want to copy the source file.

The file has the following format:

[[*user_name*@]*host_name*:][*path*/]*file_name*

user_name
Login name to be used for accessing files on the remote host. If *user_name* is not specified, the name of the user who entered the command is used.

host_name
Name of the remote host where the file is located. If *host_name* is not specified, the local host, the host from which the command was entered, is used.

path
Absolute path name or a path name relative to the login directory of the user. Shell file name expansion is not supported on either the local or remote hosts. Only single files can be copied from one host to another.

Use a backslash (\) to transfer files from a Windows host to another Windows host.

```
C:\share> lsrcp file1 hostA:c:\temp\file2
```

Use a forward slash (/) to transfer files from a UNIX host to a UNIX host.

```
lsrcp file1 hostD:/home/usr2/test/file2
```

Always use a forward slash (/) to transfer files from a UNIX host to a Windows host, or from a Windows host to a UNIX host. The operating system interprets the backslash (\) as an escape character, and **lsrcp** opens the wrong files.

For example, the following command transfers a file from UNIX to a Windows host:

```
lsrcp file1 hostA:c:/temp/file2
```

The following command transfers a file from Windows to a UNIX host:

```
C:\share> lsrcp file1 hostD:/home/usr2/test/file2
```

file_name
Name of source file. File name expansion is not supported. File names cannot include the colon character (:).

-h
Prints command usage to stderr and exits.

-V
Prints LSF release version to stderr and exits.

Examples

```
lsrcp myfile @hostC:/home/usr/dir1/otherfile
```

Copies file myfile from the local host to file otherfile on hostC.

```
lsrcp user1@hostA:/home/myfile user1@hostB:otherfile
```

Copies the file myfile from hostA to file otherfile on hostB.

```
lsrcp -a user1@hostD:/home/myfile /dir1/otherfile
```

Appends the file myfile on hostD to the file otherfile on the local host.

```
lsrcp /tmp/myfile user1@hostF:~/otherfile
```

Copies the file myfile from the local host to file otherfile on hostF in the home directory of user1.

Diagnostics

The **lsrcp** command attempts to copy source_file to target_file using RES. If RES is down or fails to copy the source_file, the **lsrcp** command uses the **rsh** command. When the -a option is specified, the **lsrcp** command uses the shell command that is specified by the **LSF_RSH** parameter in the lsf.conf file. When the -a option is not specified, the **lsrcp** command uses **rcp**.

Limitations

File transfer with the **lsrcp** command is not supported in the following contexts:

- If LSF account mapping is used, the **lsrcp** command fails when it runs under a different user account.
- LSF client hosts do not run RES, so the **lsrcp** command cannot contact RES on the submission host.
- The **lsrcp** command does not support third-party copies, when the source and target file are not on the local host. In this case, the **rcp** command or the **rsh** command (or the shell command that is specified by the **LSF_RSH** parameter in the lsf.conf file) is used. If the target file exists, the **lsrcp** command preserves the file permission modes. If the target file does not exist, the **lsrcp** command uses the source file modes consistent with the umask of the source host.

The following actions are supported:

- The **rcp** command on UNIX. If the **lsrcp** command cannot contact RES on the submission host, it attempts to use the UNIX **rcp** command to copy the file. You must set up the /etc/hosts.equiv or HOME/.rhosts file to use the **rcp** command.
- You can replace the **lsrcp** command with your own file transfer mechanism if it supports the same syntax as the **lsrcp** command. Replacing the **lsrcp** command can advantage of a faster interconnection network, or to overcome limitations with the existing **lsrcp** command. The **sbatchd** daemon looks for the **lsrcp** executable file in the LSF_BINDIR directory.

See also

rsh, **rcp**, **res**

Chapter 93. lsreghost (UNIX)

UNIX version of the **lsreghost** command registers UNIX LSF host names and IP addresses with LSF servers so that LSF servers can internally resolve these hosts without requiring a DNS server.

Synopsis

```
lsreghost -s file_path/hostregsetup
lsreghost [-h | -V]
```

Description

Directly registers UNIX LSF host names and IP addresses with LSF servers so that LSF servers can internally resolve these hosts without requiring a DNS server. The **lsreghost** command resolves the host name and IP address for LSF hosts with non-static IP addresses in environments where DNS is not able to properly resolve these hosts after their IP addresses change.

You must run this command with root privileges. If you want to register the local host at regular intervals, set up a cron job to run this command.

To use this command, the **LSF_REG_FLOAT_HOSTS=Y** parameter must be defined in the `lsf.conf` file.

Options

-s *file_path*/hostregsetup
The **lsreghost** command sends a register message to all LSF servers listed in the specified `hostregsetup` file and exits.

The `hostregsetup` file is a text file with the names of the LSF servers to which the local host must register itself. Each line in the file contains the host name of one LSF server. Empty lines and `#comment` text are ignored.

Tip: If the LSB_SHAREDIR directory, where the `reghostcache` file is located, is a shared directory that is accessible to all hosts in the cluster, you need to define only the master host in the `hostregsetup` file.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Chapter 94. lsreghost (Windows)

Windows version of the **lsreghost** command registers Windows LSF host names and IP addresses with LSF servers so that LSF servers can internally resolve these hosts without requiring a DNS server.

Synopsis

```
lsreghost -i file_path\hostregsetup
lsreghost -r | -s | -e]
lsreghost [-h | -V]
```

Description

Directly registers Windows LSF host names and IP addresses with LSF servers so that LSF servers can internally resolve these hosts without requiring a DNS server. The **lsreghost** command resolves the host name and IP address for LSF hosts with non-static IP addresses in environments where DNS is not able to properly resolve these hosts after their IP addresses change.

To use this command, the **LSF_REG_FLOAT_HOSTS=Y** parameter must be defined in the `lsf.conf` file.

Options

-e
Stops the **lsreghost** Windows service.

-r
Uninstalls and removes the **lsreghost** Windows service.

-s
Manually starts the **lsreghost** Windows service. The `lsreghost.exe` file must be in the **LSF_BINDIR** directory for the service to start.

-i *file_path*\hostregsetup
Installs **lsreghost** as a Windows service that registers the local LSF host to list of LSF servers as listed in the specified `hostregsetup` file. This file must be accessible every time the service starts up and must remain accessible while the service is running. The **lsreghost** service's startup type is set to automatic, which means that the service automatically starts up every time the local host starts up.

The `hostregsetup` file is a text file with the names of the LSF servers to which the local host must register itself. Each line in the file contains the host name of one LSF server. Empty lines and `#comment` text are ignored.

Tip: If the LSB_SHAREDIR directory (the location of the `reghostcache` file) is a shared directory that is accessible to all hosts in the cluster, you need to define only the master host in the `hostregsetup` file.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Chapter 95. lsrtasks

Displays or updates a remote task list.

Synopsis

`lsrtasks` [+ *task_name*[/*res_req*] ... | – *task_name*[/*res_req*] ...]

`lsrtasks` [`-h` | `-V`]

Description

Displays or updates a user's remote task list in the `$HOME/.lsftask` file.

When no options are specified, displays tasks that are listed in the system task file `lsf.task` and the user's task file `.lsftask`.

If the system task file `lsf.task` conflicts with the user task file, the user task file overrides the system task file.

Tasks in the remote task list are eligible for remote execution. You can associate resource requirements with each task name. Eligibility of tasks that are not specified in a task list for remote execution depends on the operation mode: local or remote. In local mode, tasks are not eligible for remote execution; in remote mode, tasks are eligible. You can specify the operation mode for deciding the eligibility of a task.

Options

+ *task_name*[/*res_req*] ...

If plus sign (+) is specified and the specified task names are not already in the user task file (`.lsftask`), adds the task names to the file with a + sign that precedes them.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a plus sign (+) or nothing, replaces the entry with the specified content; if the entry starts with a minus sign (–), deletes the entry from the `.lsftask` file.

Remote tasks can have associated resource requirements, separated by a backslash (/).

- *task_name*[/*res_req*] ...

If a minus sign (–) is specified and specified task names are not already in the user task file (`.lsftask`), adds the task names to the file with a – preceding the task name.

If any of the task names are already in the `.lsftask` file, the actual action depends on the entry in the file. If the entry starts with a –, no operation is done. If the entry starts with a +, deletes the entry from the `.lsftask` file.

Remote tasks can have associated resource requirements, separated by a backslash (/).

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Examples

```
% lsrtasks + task1 task2/"select[cpu && mem]" - task3
```

Or in restricted form:

```
% lsrtasks + task1 task2/cpu:mem - task3
```

Adds the command `task1` to the remote task list with no resource requirements, adds `task2` with the resource requirement `cpu:mem`, and removes `task3` from the remote task list.

```
% lsrtasks + myjob/swap>=100 && cpu
```

Adds `myjob` to the remote tasks list with its resource requirements.

Running **lsrtasks** with no arguments displays the resource requirements of tasks in the remote list, which is separated from the task name by a slash (/):

```
% lsrtasks
cc/cpu              cfd3d/type == SG1 && cpu  compressdir/cpu:mem
f77/cpu             verilog/cpu && cadence    compress/cpu
dsim/type == any    hspice/cpu && cadence      nas/swp > 200 && cpu
compress/-:cpu:mem  epi/hpux11 sparc           regression/cpu
cc/type == local    synopsys/swp >150 && cpu
```

Files

Reads the system task file `lsf.task`, and the user task file (`.lsftask`).

The system and user task files contain two sections, one for the remote task list, the other for the local task list. The remote tasks section starts with `Begin RemoteTasks` and ends with `End RemoteTasks`. Each line in the section is an entry that consists of a task name.

A plus sign + or a minus sign – can optionally precede each entry. If no + or – is specified, then + is assumed.

See also

lseligible, **ls_task**, **lsltasks**, `lsf.task`, **ls_eligible**

Chapter 96. lsrun

Runs an interactive task through LSF.

Synopsis

`lsrun [-l] [-L] [-P] [-S] [-v] [-m "`*host_name* ...`" | -m "`*cluster_name* ...`"] [-R "`*res_req*`"] `*command* [*argument* ...]

`lsrun [-h | -V]`

Description

With the LSF multicluster capability job forwarding model, the default is to run the task on a host in the local cluster.

By default, the **lsrun** command first tries to get resource requirement information from the remote task list to find an eligible host. Otherwise, the **lsrun** command runs the task on a host that is of the same host type (or architecture) as the submission host. If several hosts of the same architecture are available, the host with the lowest CPU and memory load is selected.

By default, if execution fails and the local host satisfies resource requirements, LSF runs the task locally.

By default, the **lsrun** command does not create a pseudo-terminal when it runs the task.

Options

-l
If execution on another host fails, runs the task locally.

-L
Forces the **lsrun** command to go through RES to run a task. By default, the **lsrun** command does not use RES if the task is going to run on the current host.

If RES execution fails and the local host satisfies resource requirements, LSF runs the task directly on local host.

-P
Creates a pseudo-terminal when the task is started on UNIX hosts. This option is necessary to run programs that require a pseudo-terminal (for example, the `vi` editor).

This option is not supported on Windows.

-S
Creates a pseudo-terminal with shell mode support when the task is started on a UNIX host. Shell mode support is required for running interactive shells or applications that redefine the **CTRL-C** and **CTRL-Z** keys (for example, **jove**).

This option is not supported on Windows.

-v
Displays the name of the host that runs the task.

-m "*host_name* ..." | -m "*cluster_name* ..."
The execution host must be one of the specified hosts. If a single host is specified, all resource requirements are ignored.

If multiple hosts are specified and you do not use the `-R` option, the execution host must satisfy the resource requirements in the remote task list (see the **lsrtasks** command). If none of the specified hosts satisfy the resource requirements, the task does not run.

With the LSF multicluster capability job forwarding model, the execution host can be a host in one of the specified clusters, if the remote cluster accepts tasks from the local cluster. Remote clusters are defined in the `RemoteClusters` section in the `lsf.cluster` file.

-R "*res_req*"
Runs the task on a host that meets the specified resource requirement. For a complete explanation of resource requirement expressions, see *Administering IBM Spectrum LSF*. To find out what resources are configured in your system, use the **lsinfo** and **lshosts** commands.

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

Exclusive resources need to be explicitly specified within the resource requirement string. For example, you defined a resource that is called `bigmem` in `lsf.shared` and defined it as an exclusive resource for `hostE` in `lsf.cluster.mycluster`. Use the following command to submit a task to run on `hostE`:

```
lsrun -R "bigmem" myjob
```

Or

```
lsrun -R "defined(bigmem)" myjob
```

If the `-m` option is specified with a single host name, the `-R` option is ignored.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

Usage

You can use the **lsrun** command together with other utility commands such as **lsplace**, **lsload**, **lsloadadj**, and **lseligible** to write load sharing applications in the form of UNIX shell scripts.

The **lsrun** command supports interactive job control. Suspending the **lsrun** command suspends both the task and the **lsrun** command. Continuing the **lsrun** command also continues the task.

If the **LSB_DISABLE_LIMLOCK_EXCL=y** parameter is configured (to enable preemption of exclusive jobs, for example), you can use the **lsrun** command to start a task on a host that is running an exclusive job.

You can simulate the `-n` option of **rsh** by redirecting input from `/dev/null`. For example,

```
lsrun cat </dev/null &
```

Diagnostics

If a problem is detected in LSF, the **lsrun** command exits with status -10 and prints an error message to `stderr`. The task does not run.

If a system call fails or incorrect arguments are specified, the exit status is -1 and an error message is printed to `stderr`.

Otherwise, the exit status is the exit status of the task.

See also

lseligible, **lshosts**, **lsload**, **lsplace**, **lsrtasks**, **ls_rexecv**, **rsh**, `lsf.cluster`

Chapter 97. lstcsh

Load sharing `tcsh` for LSF

Synopsis

`lstcsh` [*tcsh_options*] [`-L`] [*argument* ...]

Description

The **lstcsh** command is an enhanced version of the **tcsh** command. The **lstcsh** command behaves exactly like **tcsh**, except that it includes a load sharing capability with remote job execution for LSF.

By default, an **lstcsh** script is executed as a normal **tcsh** script with load sharing disabled.

If a command is eligible for remote execution, LSF selects a suitable host, and sends the command to that host. The selected host is typically a powerful or lightly loaded host that can run the command correctly.

You can restrict who can use `@` for host redirection in the **lstcsh** command with the parameter **LSF_SHELL_AT_USERS** in the `lsf.conf` file.

Remote Hosts

The **lstcsh** command provides a high degree of network transparency. Commands that are run on remote hosts behave the same as they do on the local host. The remote execution environment mirrors the local environment as closely as possible by using the same values for environment variables, terminal setup, current working directory, file creation mask, and other properties. Each modification to the local set of environment variables is automatically reflected on remote hosts.

Shell variables, nice values, and resource limits are not automatically propagated to remote hosts.

Job Control

Job control in the **lstcsh** command is the same as in the **tcsh** command except for remote background jobs. The **lstcsh** command numbers background jobs separately for each of the hosts that are used to execute them. The output of the built-in command **job** lists background jobs together with their execution hosts.

To bring a remote background job to the foreground, the host name must be specified together with an at sign (@), as in the following example:

```
fg %2 @hostA
```

Similarly, the host name must be specified when you kill a remote job. For example:

```
kill %2 @hostA
```

Options

tcsh_options
The **lstcsh** command accepts all the options that are used by the **tcsh** command. See the **tcsh** man page for the meaning of specific options.

-L
Runs a script with load sharing enabled.

Run an **lstcsh** script with load sharing enabled the following ways:

- Run the script with the `-L` option.
- Use the built-in command **source** to run the script.

- Insert **#!/local/bin/lstcsh -L** as the first line of the script (assuming you install **lstcsh** in /local/bin).

Using @ or the**lsmode** command in a script does not enable load sharing if the script was not run by using one of these three ways.

Usage

In addition to the built-in commands in the **tcsh** command, the **lstcsh** command provides the following built-in commands:

lsmode [on | off] [local | remote] [@] [v | -v] [e | -e] [t | -t] [connect [*host_name* ...]] [lsrtasks [*lsrtasks_options*]] [lsltasks [*lsltasks_options*]] [jobs]

on | off
Turns load sharing on or off. When off, you can specify @ to send a command to a remote host.

local | remote
Sets operation mode of **lstcsh**.

The mode of operation of the **lstcsh** command (local or remote) determines how the **lstcsh** command handles tasks that are not present in the remote task list nor in the local task list.

The default mode is local.

local
Local operation mode.

In local mode, a command is eligible for remote execution only if all the specified tasks are present in the remote task list in the user's tasks file $HOME/.lsftask, or if @ is specified on the command to force specified tasks to be eligible for remote execution.

Tasks in the local task list must be ran locally.

The local mode of operation is conservative, and can fail to take advantage of the performance benefits and load balancing advantages of LSF.

remote
Remote operation mode.

In this mode, a command is considered eligible for remote execution only if none of the specified tasks are present in the local task list in the user's tasks file $HOME/.lsftask.

Tasks in the remote list can be run remotely.

The remote mode of operation is aggressive, and promotes extensive use of LSF.

@
Specify @ to explicitly specify the eligibility of a command for remote execution.

The @ can be anywhere in the command line except in the first position, which is used to set the value of shell variables.

Use @ one of the following ways:

@
Specify @ followed by nothing to indicate that the command is eligible for remote execution.

@ host_name
Specify @ followed by a host name to force the command to be run on that host.

Host names and the reserved word local following @ can all be abbreviated if they do not cause ambiguity.

@ local
Specify @ followed by the reserved word local to force the command to run on the local host.

@ /res_req
Specify `@` followed by a slash (/) and a resource requirement string to indicate that the command is eligible for remote execution, and that the specified resource requirements must be used instead of the requirements in the remote task list.

When you specify resource requirements after the `@`, you can use / only if the first requirement characters you specified are the first characters of a host name.

e | -e
Turns on verbose eligibility mode (e) or off (`-e`).

When eligibility verbose mode is on, the **lstcsh** command shows whether the command is eligible for remote execution, and displays the resource requirement that is used if the command is eligible.

The default is off.

v | -v
Turns on task placement verbose mode (v) or off (`-v`). When verbose mode is on, the **lstcsh** command displays the name of the host on which the command runs if the command does not run on the local host.

The default is on.

t | -t
Turns on wall clock timing (`t`) or off (`-t`).

When timing is on, the actual response time of the command is displayed. The response time is the total elapsed time in seconds from the time you submit the command to the time the prompt comes back.

This time includes all remote execution usage. The **csh** time built-in does not include the remote execution usage.

Wall clock timing is an impartial way of comparing the response time of jobs that are submitted locally or remotely because all the load sharing usage is included in the displayed elapsed time.

The default is off.

connect [*host_name* ...]
Establishes connections with specified remote hosts. If no hosts are specified, lists all the remote hosts to which an **lstcsh** connection was established.

A plus sign (+) with a remote host indicates that a server-shell was also started on it.

lsrtasks [+ *task_name*[/*res_req* ...] | - *task_name*[/*res_req* ...]]
Displays or update a user's remote task list in the user's task list `$HOME/.lsftask`.

The **lsrtasks** command under the **lstcsh** command has the same function as the external command **lsrtasks**, except that the modified remote task list takes effect immediately for the current **lstcsh** session.

See the **lsrtasks** command for more details.

lsltasks [+ *task_name* ... | - *task_name* ...]

Displays or update a user's local task list in the user's task list `$HOME/.lsftask`.

The **lsltasks** command under the **lstcsh** command has the same function as the external command **lsltasks**, except that the modified local task list takes effect immediately for the current **lstcsh** session.

See the **lsltasks** command for more details.

jobs
Lists background jobs together with the execution hosts to give you more control over your background jobs.

Files

The **lstcsh** command uses three optional configuration files:

- `.shrc` is used by **lstcsh** alone.
- `.hostrc` is used by **lstcsh** alone.
- `.lsftask` determines general task eligibility.

~/.shrc
Use this file when you want an execution environment on remote hosts that is different from the environment on the local host. This file is sourced automatically on a remote host when a connection is established. For example, if the remote host is of different type, you might need to run a version of the executable file for that particular host type. It might be necessary to set a different path on the remote host.

~/.hostrc
Use this file to indicate a list of host names to which the user wants to be connected (asynchronously in the background) at **lstcsh** startup time. This file saves the time that is spent in establishing the connections dynamically during execution of shell commands. After a connection is set up, you can run further remote commands on those connected hosts with little processing load.

~/.lsftask
Use this file to specify lists of remote and local tasks that you want to be added to the respective system default lists. Each line of this file is of the form *task_name*`/`*res_req*, where *task_name* is the name of a task, and *res_req* is a string that specifies the resource requirements of the task. If *res_req* is not specified, the command runs on hosts of the same type as the local host.

Limitations

Type ahead for the **next** command is discarded when a job is running in the foreground on a remote host.

You cannot provide input data to load sharing shell scripts (that is, shell scripts with load-shared content).

The **lstcsh** command is fully compatible with **tcsh** version 6.03 7-bit mode. Any feature that is not included in **tcsh** 6.03 is not supported.

See also

csh, **tcsh**, **lsrtasks**, **lsltasks**, **lseligible**, **lsinfo**, **lsload**

Chapter 98. pam

Parallel Application Manager – job starter for MPI applications

HP-UX vendor MPI syntax
`bsub pam -mpi mpirun` [*mpirun_options*] *mpi_app* [*argument* ...]

Generic PJL framework syntax
`bsub pam` [`-t`] [`-v`] [`-n` *num_tasks*] `-g` [*num_args*] *pjl_wrapper* [*pjl_options*] *mpi_app* [*argument* ...] `pam` [`-h`] `pam` [`-V`]

Description

The Parallel Application Manager (PAM) is fully integrated with LSF. PAM acts as the supervisor of a parallel LSF job.

MPI jobs started by the **pam** command can be submitted only through batch jobs, PAM cannot be used interactively to start parallel jobs. The **sbatchd** daemon starts PAM on the first execution host.

PAM has the following functionality for all parallel application processes (tasks):

- Uses a vendor MPI library or an MPI Parallel Job Launcher (PJL), for example, **mpirun** or **poe**, to start a parallel job on a specified set of hosts in an LSF cluster.
- PAM contacts RES on each execution host that is allocated to the parallel job.
- PAM queries RES periodically to collect resource usage for each parallel task and passes control signals through RES to all process groups and individual running tasks, and cleans up tasks as needed.
- Passes job-level resource usage and process IDs (PIDs and PGIDs) to **sbatchd** for enforcement
- Collects resource usage information and exit status upon termination

Task startup for vendor MPI jobs

The **pam** command starts a vendor MPI job on a specified set of hosts in an LSF cluster. The **pam** command that starts an MPI job requires the underlying MPI system to be LSF-aware, using a vendor MPI implementation that supports LSF (for example, HP-UX vendor MPI).

PAM uses the vendor MPI library to create the child processes needed for the parallel tasks that make up your MPI application. It starts these tasks on the systems that are allocated by LSF. The allocation includes the number of execution hosts needed, and the number of child processes needed on each host.

Task startup for generic PJL jobs

For parallel jobs submitted with **bsub**:

- PAM starts the PJL, which in turn starts the TaskStarter (TS).
- TS starts the tasks on each execution host, reports the process ID to PAM, and waits for the task to finish.

Two environment variables enable PAM to run scripts or binary files before or after PAM is started. These variables are useful if you customize the `mpirun.lsf` script and have job scripts that call the `mpirun.lsf` script more than once.

$MPIRUN_LSF_PRE_EXEC
Runs before PAM is started.

$MPIRUN_LSF_POST_EXEC
Runs after PAM is started.

Options for vendor MPI jobs

-auto_place
The `-auto_place` option on the **pam** command line tells the IRIX **mpirun** library to start the MPI application according to the resources allocated by LSF.

-mpi
On HP-UX, you can have LSF manage the allocation of hosts to achieve better resource usage by coordinating the start-up phase with the **mpirun** command. Precede the regular MPI **mpirun** command with the following command:

```
bsub pam -mpi
```

For HP-UX vendor MPI jobs, the `-mpi` option must be the first option of the **pam** command.

For example, the following **mpirun** command runs a single-host job:

```
mpirun -np 14 a.out
```

To have LSF select the host, include the **mpirun** command in the **bsub** job submission command:

```
bsub pam -mpi mpirun -np 14 a.out
```

-n *num_tasks*
The number of processors that are required to run the parallel application, typically the same as the number of parallel tasks in the job. If the host is a multiprocessor, one host can start several tasks.

You can use both the **bsub -n** and **pam -n** commands in the same job submission. The number that is specified in the **pam -n** option must be less than or equal to the number specified by the **bsub -n** command. If the number of tasks that are specified with the **pam -n** command is greater than the number that is specified by the **bsub -n** command, the **pam -n** command is ignored.

For example, you can specify the following command:

```
bsub -n 5 pam -n 2 -mpi -auto_place a.out
```

The job requests five processors, but PAM starts only two parallel tasks.

***mpi_app* [*argument* ...]**
The name of the MPI application to be run on the listed hosts. This name must be the last argument on the command line.

-h
Prints command usage to `stderr` and exit.

-V
Prints LSF release version to `stderr` and exit.

Options for generic PJL jobs

-t
This option tells the **pam** command not to print the MPI job tasks summary report to the standard output. By default, the summary report prints the task ID, the host that it ran on, the command that was run, the exit status, and the termination time.

-v
Verbose mode. Displays the name of the execution host or hosts.

-g [*num_args*] *pjl_wrapper* [*pjl_options*]
The `-g` option is required to use the generic PJL framework. You must specify all the other **pam** options before `-g`.

num_args
Specifies how many space-separated arguments in the command line are related to the PJL (after that, the remaining section of the command line is assumed to be related to the binary application that starts the parallel tasks).

pjl_wrapper
The name of the PJL.

pjl_options
Optional arguments to the PJL.

For example:

- A PJL named `no_arg_pjl` takes no options, so *num_args*=1. The syntax is:

```
pam [pam_options] -g 1 no_arg_pjl job [job_options]
```

- A PJL is named **3_arg_pjl** and takes the options `-a`, `-b`, and *group_name*, so *num_args*=4. Use the following syntax:

```
pam [pam_options] -g 4 3_arg_pjl -a -b group_name job [job_options]
```

-n *num_tasks*
The number of processors that are required to run the MPI application, typically the number of parallel tasks in the job. If the host is a multiprocessor, one host can start several tasks.

You can use both the **bsub -n** and **pam -n** commands in the same job submission. The number that is specified in the **pam -n** option must be less than or equal to the number specified by the **bsub -n** option. If the number of tasks that are specified with the **pam -n** option is greater than the number specified by the **bsub -n** option, the **pam -n** option is ignored.

***mpi_app* [*argument* ...]**
The name of the MPI application to be run on the listed hosts. This name must be the last argument on the command line.

-h
Prints command usage to `stderr` and exit.

-V
Prints LSF release version to `stderr` and exits.

Exit Status

The **pam** command exits with the exit status of the **mpirun** command or the PJL wrapper.

See also

bsub

Chapter 99. patchinstall

UNIX only. Manage patches in LSF cluster.

Synopsis

`patchinstall` [`-f` *env_file*] [`--silent`] *package* ...

`patchinstall -c` [`-f` *env_file*] [`--silent`] *package* ...

`patchinstall -r` [`-f` *env_file*] [`--silent`] *package*

`patchinstall -r` [`-f` *env_file*] [`--silent`] *build_number*

`patchinstall -h`

Description

Permission that is required to run this command depends on the package contents and the original cluster installation account. You normally log on as root to run the **patchinstall** command, but you can patch some binary files as cluster administrator (`lsfadmin`).

By default, the command installs one or more packages in an existing cluster.

The cluster location is normally determined by your environment setting, so ensure that your environment is set before you run this command (for example, you sourced the `cshrc.lsf` or `profile.lsf` file).

Specify the packages that you want to install.

The installer does some checking first. If it does not find a problem, it prompts you to proceed with installation. If you confirm, it backs up the current binary files to the patch backup directory and then installs the specified packages on the cluster, updating or adding new binary files. It does not modify any existing configuration files. If the installation of a package has problems, it automatically rolls back to the cluster's previous state. It records the changes in the patch history directory. This additional checking can take more time than installing with **lsfinstall**.

The command can also be used to check or roll back a patch:

Check
Do the checking for the packages without installing them. For more information, see the `-c` option.

Roll back
Remove the most recent patch and return the cluster to the previous patch level. To roll back multiple versions, you must roll back one patch level at a time, in the reverse order of installation. For more information, see the `-r` option.

If you want to enable installation through AFS, take the following steps.

1. Modify **patchinstall** to set the environment.

   For the following original settings:

   ```
   CHOWN="chown"
   CHMOD="chmod"
   IGNORECHECKFILEOWNER="n"
   ```

   Set the following environment:

   ```
   CHOWN="asudo chown"
   CHMOD="asudo chmod"
   IGNORECHECKFILEOWNER="y"
   ```

2. Create an environment file where all LSF paths point to the `volrw` instead of `volro`.
3. Run the following command:

```
patchinstall -f environment_file package
```

The path to *environment_file* must include the correct LSF_TOPDIR installation directory.

Options

-c
Check as if to install, but do not proceed with installation.

Specify each package that you want to check. You can specify multiple packages.

Checks that the existing cluster is compatible with the patch (the same version of the product is already installed on the same binary types). Fixes and fix packs might also require installation of a specific enhancement pack.

Checks that your user account has permission to write to the installation directory, backup directory, and history directory.

Lists existing files that are overwritten by the patch.

Lists files that to be added by the patch.

-f *env_file*
Use This option only if you cannot set your environment (for example, you cannot source `cshrc.lsf` or `profile.lsf`).

Specify the full path and file name of a file (such as your LSF `install.config` file) that properly defines the parameter LSF_TOP.

If you use this option, the command gets the cluster location from this file, not from the settings in your environment.

-h
Outputs command usage and exits.

-r
Rollback. You must specify the most recently installed patch. The installer checks all binary types and finds all instances where the most recently installed patch has the same build number. These packages are removed and the cluster reverts to the previous patch level.

Specify the build number of the most recent patch or specify full path to the package you used to install the most recent patch. The installer automatically checks the package to determine the build. You cannot specify any other build.

To remove multiple patches and roll back multiple versions, you must run the command multiple times and roll back one patch level at a time.

You cannot roll back if the backup files from the previous patch level are unavailable (if you deleted them from the patch backup directory).

--silent
Silent mode. Install or roll back without any interactive prompts for confirmation.

Output

Status information and prompts are displayed in your command console.

When you patch or roll back the cluster, status information is logged to the `patch.log` file. When you check a package, status information is logged to the `precheck.log` file.

If any problems are found when you check a package, errors are displayed in your command console and also logged to the `patch.err` file.

See also

`pversions` command
Displays the patch level of products that are installed in your cluster.

`install.config` file
Defines the parameter **LSF_TOP**.

`patch.conf` file
Defines backup and history directories.

Chapter 100. pversions (UNIX)

UNIX version of the command. Displays the version information for IBM Spectrum LSF installed on UNIX hosts.

Synopsis

`pversions` [`-f` *env_file*]

`pversions -p` [`-f` *env_file*] *product_name*

`pversions -b` [`-f` *env_file*] *build_number*

`pversions -q` [`-f` *env_file*] *file_name*

`pversions -c` *package_name*

`pversions -h`

Description

By default, displays the version and patch level of IBM Spectrum LSF.

The cluster location is normally determined by your environment setting, so ensure that your environment is set before you run this command (for example, you sourced either the **`cshrc.lsf`** or the **`profile.lsf`** file).

For each binary type, displays basic version information (package build date, build number, package installed date) and lists patches installed (package type, build number, date that is installed, fixes).

Optionally, the command can also be used to make the following checks:

- Check the contents of a package before it is installed.
- Show information about a specific LSF installation.
- Show information about installed packages from specific build.
- Find current versions of a specific file and see information for each file.

Options

-f *env_file*

Use this option only if you cannot set your environment (for example, you cannot source the **`cshrc.lsf`** or **`profile.lsf`** file).

Specify the full path and file name of a file (such as your LSF `install.config` file) that defines the parameter LSF_TOP.

If you use this option, the command gets the cluster location from this file, not from the settings in your environment.

-b build_number

Specify the build number of an installed patch. You can specify the most recent full installation or patches installed after the most recent full installation.

Displays information and the contents of the build (binary type and install date, notes, fixes, and files in the package).

-c package_name

Specify the full path and file name of an uninstalled package. For this option, you do not need to set your environment because a cluster is not required.

Displays package contents (notes, fixes, and files in the patch).

-p product_name

Specify one LSF product to see information for that product only. Specify LSF to see information about LSF.

-q file_name

Specify the file name of one installed file.

For each binary type, displays basic version information and file location. If the binary was updated after the most recent full installation, displays additional information about the most recent patch that updated the file (build number, fixes, notes, date installed).

-h

Outputs command usage and exits.

Output

Information is displayed in your command console.

Product version information (Default)

By default, displays product information for entire cluster.

For each product, displays product name and version followed by specific information about each binary type.

For each binary type, displays basic version information (package build date, build number, package installed date) and lists any patches that are installed (package type, build number or fix number, date installed).

binary type
Binary type, build number of binary, and build date of the binary for the most recent full installation. A full installation is installation of any distribution that contains a complete set of new binary files. A full installation can be a new cluster, an upgrade, or patching with an enhancement pack.

installed
Date the binary was installed for the most recent full installation.

patched
For each patch after the most recent full installation, displays fix number, build number, and date patch was installed. If the patch was a fix pack, multiple fixes are listed.

File version information (-q)

The -q option displays information for specified file only.

For each product that contains the specified file, displays product name and version followed by specific information about each binary type.

For each binary type that contains the specified file, displays basic version information and file location. If the binary was updated after the most recent full installation, displays additional information about the most recent patch that updated the file (build number, fixes, notes, date installed).

binary type
Binary type, build number of binary, and build date of the binary for the most recent full installation. A full installation is any distribution that contains a complete set of new binary files. A full installation can be a new cluster installation, a version upgrade, or patching with an enhancement pack.

installed
Date the binary was installed for the most recent full installation.

file
Full path to the version of the file that is used for this binary type.

last patched
For the last patch to update the file after the most recent full installation, displays build number and date patch was installed.

last patch notes
Optional. Some information that is provided for the last patch that updated the file.

last patch fixes
Fixes that are included in the last patch that updated the file.

Build version information (-b)

The -b option displays information for patches with the specified build number only.

For each product, if the product is using binary files from the specified build, displays product name and version followed by specific information about each binary type.

For each binary type, displays the following information:

binary type
Binary type, build number and build date of the patch.

installed
Date the patch was installed.

notes
Optional. Some information that is provided for the build.

fixes
Fixes that are included in the patch.

files
Files that are included in the patch. Not shown for a full distribution such as enhancement pack. Full path to the file installed by this patch.

Package version information (-c)

The -c option displays version information for a specified uninstalled package.

product
Displays product name and version.

binary type
Binary type, build number and build date of the patch.

notes
Optional. Some information that is provided for the build.

fixes
Fixes that are included in the patch.

files
Files that are included in the patch (not shown for a full distribution such as enhancement pack). Relative path to the file.

Chapter 101. pversions (Windows)

Windows version of the command. Displays the version information for IBM Spectrum LSF installed on a Windows host.

Synopsis

`pversions` [*product_name*]

```
pversions -h
```

```
pversions -V
```

Description

Displays the version and patch level for LSF installed on a Windows host, and the list of patches installed.

Options

product_name

Specify the product for which you want version information. Specify one of the following product names:

EGO

To see version information for enterprise grid orchestrator.

LSF

To see version information for LSF.

Symphony

To see version information for IBM Spectrum Symphony.

-h

Prints command usage to `stderr` and exits from the software.

-V

Prints product version to `stderr` and exits.

Chapter 102. rclogsvalidate

Displays audit logs for LSF resource connector.

Synopsis

`rclogsvalidate -l [-d start_tine,end_time]`

`rclogsvalidate -n number`

`rclogsvalidate [-h]|[-V]`

Description

Options

-l [-d *start_time*,*end_time*]
Displays audit logs for the last hour.

To display audit logs for a specified time interval, use the `-d` option and specify the times in the following format: *yyyy*/*mm*/*dd*/*HH*:*MM*

Use a comma to separate the start and end times and do not use spaces when specifying the time interval.

-n *number*
Displays and checks audit logs in consecutive order starting with the specified number (`rc.audit.`*number*) and ending with the current log. You must specify a positive integer.

If none of the checked logs is tampered, this command exits with return code 0.

If any logs are tampered, this command stops and exits with return code 99.

This command returns an error if the `rc.audit.`*integer* file (from `rc.audit.`*number* to the current log file) does not exist.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF resource connector release version to `stderr` and exits.

Chapter 103. ssacct

Displays accounting statistics about finished LSF session scheduler jobs.

Synopsis

`ssacct [-l] job_ID [task_ID | "task_ID[index]"]`

`ssacct [-l] "job_ID [index]" [task_ID | "task_ID[index]"]`

`ssacct [-l] -f log_file [job_ID [task_ID | "task_ID[index]"]]`

`ssacct [-l] -f log_file ["job_ID [index]" [task_ID | "task_ID[index]"]]`

`ssacct [-h] | [-V]`

Description

By default, displays accounting statistics for all finished jobs that are submitted by the user who used the command.

Options

-l
Long format. Displays extra accounting statistics.

-f *log_file*
Searches the specified job log file for accounting statistics. Specify either an absolute or relative path.

By default, the **ssacct** command searches for accounting files in directory that is defined by the **SSCHED_ACCT_DIR** parameter in the `lsb.params` file. Use this option to parse a specific file in a different location. You can specify a log file name, or a job ID, or both a log file and a job ID. The following commands are correct:

```
ssacct -f log_file job_ID
```

```
ssacct -f log_file
```

```
ssacct job_ID
```

The specified file path can contain up to 4094 characters for UNIX, or up to 255 characters for Windows.

***job_ID* | "*job_ID*[*index*]"**
Displays information about the specified jobs or job arrays.

***task_ID* | "*task_ID*[*index*]"**
Displays information about the specified tasks or task arrays.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF session scheduler release version to `stderr` and exits.

Output: default format

Statistics on all tasks in the session. The following fields are displayed:

- Total number of done tasks
- Total CPU time in seconds used
- Average CPU time in seconds used

- Maximum CPU time in seconds of a task
- Minimum CPU time in seconds of a task
- Total wait time in seconds
- Average wait time in seconds
- Maximum wait time in seconds
- Minimum wait time in seconds
- Average turnaround time (seconds/task)
- Maximum turnaround time (seconds/task)
- Minimum turnaround time (seconds/task)
- Average hog factor of a job (CPU time/turnaround time)
- Maximum hog factor of a task (CPU time/turnaround time)
- Minimum hog factor of a task (CPU time/turnaround time)

The total, average, minimum, and maximum statistics are on all specified tasks.

The wait time is the elapsed time from job submission to job dispatch.

The turnaround time is the elapsed time from job submission to job completion.

The hog factor is the amount of CPU time that is consumed by a job, divided by its turnaround time.

Output: long format with the -l option

In addition to the fields displayed by default in the SUMMARY output, the `-l` option displays the following fields:

CPU_T
CPU time in seconds used by the task.

WAIT
Wall clock time in seconds between when the task was submitted to the LSF session scheduler and when it was dispatched to an execution host.

TURNAROUND
Wall clock time in seconds between when the task was submitted to the LSF session scheduler and when it completed running.

STATUS
Status that indicates the job was either successfully completed (done) or exited (exit)

HOG_FACTOR
Average hog factor, equal to CPU time/turnaround time.

Example: default format

```
ssacct 108 1[1]
Accounting information about tasks that are:
  - submitted by all users.
  - completed normally or exited.
  - executed on all hosts.
------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done tasks:       1      Total number of exited tasks:     0
 Total CPU time consumed:       0.0      Average CPU time consumed:     0.0
 Maximum CPU time of a task:     0.0      Minimum CPU time of a task:     0.0
 Total wait time:     2.0
 Average wait time:    2.0
 Maximum wait time:    2.0      Minimum wait time:    2.0
 Average turnaround time:         3 (seconds/task)
 Maximum turnaround time:         3      Minimum turnaround time:         3
 Average hog factor of a task:  0.01 ( cpu time / turnaround time )
 Maximum hog factor of a task :  0.01      Minimum hog factor of a task:  0.01
```

Example: long format with the -l option

```
ssacct -l 108 1[1]
Accounting information about tasks that are:
  - submitted by all users.
  - completed normally or exited.
  - executed on all hosts.

------------------------------------------------------------------------
Job <108>, Task <1>, User <user1>, Status <Done> Command <myjob>
Thu Nov  1 13:48:03 2008: Submitted from host <hostA>;
Thu Nov  1 13:48:05 2008: Dispatched to <hostA>, Execution CWD </home/user1/src>
Thu Nov  1 13:48:06 2008: Completed <done>.

Accounting information about this job:
     CPU_T     WAIT     TURNAROUND   STATUS     HOG_FACTOR
      0.03        2              3     done         0.0113
------------------------------------------------------------------------

SUMMARY:      ( time unit: second )
 Total number of done tasks:       1      Total number of exited tasks:     0
 Total CPU time consumed:       0.0      Average CPU time consumed:     0.0
 Maximum CPU time of a task:     0.0      Minimum CPU time of a task:     0.0
 Total wait time:     2.0
 Average wait time:    2.0
 Maximum wait time:    2.0      Minimum wait time:    2.0
 Average turnaround time:         3 (seconds/task)
 Maximum turnaround time:         3      Minimum turnaround time:         3
 Average hog factor of a task:  0.01 ( cpu time / turnaround time )
 Maximum hog factor of a task :  0.01      Minimum hog factor of a task:  0.01
```

Files

Reads *job_ID*.ssched.acct

See also

ssched, lsb.params

Chapter 104. ssched

Submit tasks through LSF session scheduler.

Synopsis

`ssched` [*options*] *command*

`ssched` [*options*] `-tasks` *task_definition_file*

`ssched` [*options*] `-tasks` *task_definition_file command*

`ssched` [`-h` | `-V`]

Description

Options can be specified on the **ssched** command line or on a line in a task definition file. If the option is specified on the command line, it applies to all tasks, whether they are specified on the command line or in a file. Options that are specified in a file apply only to the command on that line. Options in the task definition file override the same option that is specified on the command line.

ssched exit codes

0
All tasks completed normally.

1
An unspecified error occurred.

3
All tasks completed, but some tasks have a nonzero exit code.

4
Error parsing **ssched** command line parameters or tasks definition file. No tasks were run.

5
Exceeded the limit that is specified by the **SSCHED_MAX_TASKS**parameter.

Task Definition File Format

The task definition file is an ASCII file. Each line represents one task, or an array of tasks. Each line has the following format:

```
[task_options] command [arguments]
```

Command options

-1 | -2 | -3
Enables increasing amounts of debug output.

-C
Check all parameters and the task definition file. Exit immediately after the check is complete. An exit code of 0 indicates that no errors were found. Any nonzero exit code indicates an error. The **ssched -C** command can be run outside of LSF.

-p

Do not delete the temporary working directory. This option is useful when you diagnose errors.

Task options

-E "*pre_exec_command* [*arguments* ...]"
Runs the specified job-based pre-execution command on the execution host before running the task.

The task pre-execution behavior mimics the behavior of LSF job pre-execution. However, the task pre-execution command cannot run as root.

The standard input and output for the pre-execution command are directed to the same files as the job. The pre-execution command runs under the same user ID, environment, home, and working directory as the job. If the pre-execution command is not in the user's usual execution path (the $PATH variable), the full path name of the command must be specified.

-Ep "*post_exec_command* [*arguments* ...]"
Runs the specified job-based post-execution command on the execution host after the task finishes.

The task post-execution behavior mimics the behavior of LSF job post-execution. However, the task post-execution command cannot run as root.

If the post-execution command is not in the user's usual execution path (the $PATH variable), the full path name of the command must be specified.

-e *error_file*
Specify a file path. Appends the standard error output of the job to the specified file.

If the parameter **LSB_STDOUT_DIRECT** in the `lsf.conf` file is set to Y or y, the standard error output of a task is written to the file you specify as the task runs. If the **LSB_STDOUT_DIRECT** parameter is not set, standard error output of a task is written to a temporary file and copied to the specified file after the task finishes.

You can use the special characters %J, %I, %T, and %X in the name of the output file.

%J
Replaced by the job ID.

%I
Replaced by the job array index.

%T
Replaced with the task ID.

%X
Replaced by the task array index.

If the current working directory is not accessible on the execution host after the job starts, Session Scheduler writes the standard error output file to `/tmp/`.

Note: The file path can contain up to 4094 characters, including the directory, file name, and expanded values for%J, %I, %T, and %X.

-i *input_file*
Gets the standard input for the job from specified file. Specify an absolute or relative path. The input file can be any type of file, though it is typically a shell script text file.

If `-i` is not specified, standard input defaults to `/dev/null`.

You can use the special characters %J, %I, %T, and %X in the name of the input file.

%J
Replaced by the job ID.

%I
Replaced by the job array index.

%T
Replaced with the task ID.

%X
Replaced by the task array index.

Note: The file path can contain up to 4094 characters, including the directory, file name, and expanded values for%J, %I, %T, and %X.

-J *task_name*[*index_list*]
Specifies the indices of the task array. The index list must be enclosed in square brackets. The index list is a comma-separated list whose elements have the syntax *start*[-*end*[:*step*]], where *start*, *end* and step are positive integers. If the step is omitted, a step of one is assumed. The task array index starts at one.

All tasks in the array have the same option parameters. Each element of the array is distinguished by its array index.

-j "*starter* [*starter*] ['%USRCMD'] [*starter*]"
Task job starter. Creates a specific environment for submitted tasks before execution.

The job starter is any executable file that can be used to start the task (that is, it can accept the task as an input argument). Optionally, more strings can be specified.

By default, the user commands run after the job starter. A special string, %USRCMD, can be used to represent the position of the user's task in the job starter command line. The %USRCMD string can be followed by more commands.

-o *output_file*
Specify a file path. Appends the standard output of the task to the specified file. The default is to output to the same stdout as the **ssched** command.

If only a file name is specified, LSF writes the output file to the current working directory. If the current working directory is not accessible on the execution host after the task starts, LSF writes the standard output file to /tmp/.

If the parameter **LSB_STDOUT_DIRECT** in the lsf.conf file is set to Y or y, the standard output of a task is written to the file you specify as the task runs. If the **LSB_STDOUT_DIRECT** parameter is not set, standard output is written to a temporary file and copied to the specified file after the task finishes.

You can use the special characters %J, %I, %T, and %X in the name of the output file.

%J
Replaced by the job ID.

%I
Replaced by the job array index.

%T
Replaced with the task ID.

%X
Replaced by the task array index.

Note: The file path can contain up to 4094 characters, including the directory, file name, and expanded values for %J, %I, %T, and %X.

-M *mem_limit*
Sets a per-process (soft) memory limit for all the processes that belong to the task.

By default, the limit is specified in KB. Use the **LSF_UNIT_FOR_LIMITS** parameter in the lsf.conf file to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

Set a task level memory limit only if it less than the job limit.

-Q "*exit_code ...*"
Task requeue exit values. Enables automatic task requeue and sets the **LSB_EXIT_REQUEUE** environment variable. Use spaces to separate multiple exit codes. The output from the failed run is not saved, and the user is not notified by LSF.

-W [*minutes*:]*seconds*
Sets the runtime limit of the task. If a task runs longer than the specified run limit, the task is sent a SIGKILL signal.

The run limit is in the form [*minutes*:]*seconds*. The seconds can be specified as a number greater than 59. For example, three and a half minutes can either be specified as 3:30, or 210. The run limit you specify is the absolute run time.

-tasks *task_definition_file*
Specify tasks through a task definition file.

***command* [*argument*]**
The command can be anything that is provided to a UNIX Bourne shell. The command is assumed to begin with the first word that is not part of an option. All arguments that follow command are provided as the arguments to the command.

The job command can be up to 4094 characters long.

-h
Prints command usage to stderr and exits.

-V
Prints LSF session scheduler release version to stderr and exits.

See also

ssacct, lsb.params

Chapter 105. taskman

Checks out a license token and manages interactive UNIX applications.

Synopsis

`taskman -R "rusage[`*token=number* `[:duration=`*minutes* | *hours* h]
`[:`*token=number*`[:duration=`*minutes* | *hours* h] `[| |` *token=number*`[:duration=`*minutes* | *hours* h]
`[:`*token=number* `[:duration=`*minutes* | *hours* h]`]` ...] `[-Lp` *project*`] [-N` *n_retries*`] [-v]` *command*

`taskman [-h | -V]`

Description

Runs the interactive UNIX application on behalf of the user. When it starts, the task manager connects to LSF License Scheduler to request the application license tokens. When all the requested licenses are available, the task manager starts the application. While the application is running, the task manager monitors resource usage, CPU, and memory, and reports the usage to LSF License Scheduler. When the application ends, the task manager exits.

By default, a license is reserved for the duration of the task, so the application can check out the license at any time. Use the `duration` keyword to reallocate unused licenses if the application fails to check out the license before the reservation expires.

By default, the **taskman** command does not detect or use batch resources. To allow the **taskman** command to detect batch resources, set **ENABLE_INTERACTIVE=y** parameter in the `lsf.licensescheduler` file.

Options

command
Required. The command to start the job that requires the license.

-v
Verbose mode. Displays detailed messages about the status of configuration files.

-N *n_retries*
Specifies the maximum number of retry attempts that the **taskman** command makes to connect to the daemon. If this option is not specified, the **taskman** command retries indefinitely.

-Lp *project*
Optional. Specifies the interactive license project that is requesting tokens. The client must be known to LSF License Scheduler.

License project limits do not apply to the **taskman** command jobs even with `-Lp` specified.

-R rusage[*token=number* [:duration=*minutes* | *hours* h] [:*token=number* [:*duration=minutes* | *hours* h] [|| *token=number* [:*duration=minutes* | *hours* h] [:*token=number* [:*duration=minutes* | *hours* h]]] ...]
Required. Specifies the type and number of license tokens to request from LSF License Scheduler. Optionally, specifies a time limit for the license reservation, expressed as an integer (the keyword h following the number indicates hours instead of minutes). You can specify multiple license types, with different duration values. Separate each requirement with a colon (`:`) as a logical AND operator, and a double-pipe (`||`) as a logical OR operator. Enclose the entire list in one set of square brackets.

Note: If you specify alternative or compound resource requirements, the **taskman** command accepts only the first resource requirement string and ignores the other resource requirement strings.

For example,

Alternative resource requirement

```
taskman -R "{rusage[f2=2]}||{rusage[f2=1]}" myjob
```

Compound resource requirement

```
taskman -R "{rusage[f2=2]}+{rusage[f2=1]}" myjob
```

In both cases, the **taskman** command accepts only the `rusage[f2=2]` string.

-h

Prints command usage to `stderr` and exits.

-V

Prints the LSF License Scheduler release version to `stderr` and exits.

Chapter 106. tspeek

Displays the `stdout` and `stderr` output of an unfinished Terminal Services job.

Synopsis

`tspeek` *job_ID*

`tspeek` [`-h` | `-V`]

Description

Displays the standard output and standard error output that was produced by one of your unfinished Terminal Services jobs, up to the time that this command is started.

This command is useful for monitoring the progress of a job and identifying errors. If errors are observed, valuable user time and system resources can be saved by terminating an erroneous job.

The **tspeek** command is supported on Windows and Linux. You cannot use the **tspeek** commands to monitor job output from UNIX. The **tspeek** command on Linux requires the **rdesktop** application.

You can use the **tspeek** command from any Linux host where the **rdesktop** application is installed to view the output of a Terminal Services job. For example, if your job ID is 23245, run the following command:

```
tspeek 23245
```

Options

job_ID
Operates on the specified Terminal Services job.

-h
Prints command usage to `stderr` and exits.

-V
Prints LSF release version to `stderr` and exits.

See also

tssub

Chapter 107. tssub

Submits a Terminal Services job to LSF.

Synopsis

`tssub` [*bsub_options*] *command* [*arguments*]

`tssub` [`-h` | `-V`]

Description

Submits a Terminal Services job for batch execution and assigns it a unique numerical job ID.

The **tssub** command is a wrapper around the **bsub** command that submits jobs only to hosts that have Microsoft Terminal Services installed. For **bsub** command options, see the **bsub** command.

You submit Terminal Services job with the **tssub** command instead of the **bsub** command. If the terminal window is closed, the job remains running. You can reconnect to view the job with the **tspeek** command.

The **tssub** command is supported on Windows and Linux. You cannot use the **tssub** command to submit Terminal Services jobs from UNIX.

If the job is dispatched to a host in which Terminal Services is not installed or properly configured, the job is set to the PEND state. A pending reason is written to the `sbatchd.log.`*`host_name`* file.

If the **tssub -I** command is specified, a terminal display is visible on the submission host after the job starts.

If the job is not a GUI job, LSF runs a command window and output is displayed in the command window when something is written to `stdout`.

Pre- and post-execution commands are run within the terminal session. The job does not complete until post-execution commands complete.

If you use the **bjobs -l** command to monitor the job, you see the following message:

```
"External Message 2 was posted from LSF\lsfadmin to message box 2"
```

The body of the message contains the ID of the terminal session that was created.

Use the **tspeek** command to view job output.

The **tssub** command sets the **LSB_TSJOB** and **LSF_LOGON_DESKTOP** environment variables. These variables are then transferred to the execution host:

LSF_LOGON_DESKTOP
When **LSF_LOGON_DESKTOP=1**, jobs run in interactive foreground sessions, and GUIs are displayed on the execution host. If this parameter is not defined, jobs run in the background.

LSB_TSJOB
When the **LSB_TSJOB** variable is defined to any value, it indicates to LSF that the job is a Terminal Services job.

Limitations

- You cannot use the **bmod** command to modify a job that is submitted as a Terminal Services job to become a non-Terminal Services job.
- The **bsub -o** *out_file* option is not supported for the **tssub** command.
- Only Windows **bsub** command options are supported for the **tssub** command. For example, you cannot use the **bsub** options `-Ip`, `-Is`, `-L` *login_shell* with the **tssub** command.
- Interactive **bsub** options (`-I`, `-Ip`, `-Is`) are not supported by the **tssub** command on Linux.

- If user mapping is defined, the user who runs the **tspeek** command must have the required privileges to access the session.
- LSF multicluster capability is not supported.

Options

bsub_options

Only Windows **bsub** options are supported for the **tssub** command. For example, you cannot use the **bsub** options `-Ip`, `-Is`, `-L` *login_shell* with the **tssub** command.

For **bsub** options, see the **bsub** command.

***command* [*argument*]**

The job can be specified by a command line, or through the standard input if the command is not present on the command line. The *command* is assumed to begin with the first word that is not part of a **tssub** option. All arguments that follow *command* are provided as the arguments to the *command*.

The job command can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows. If no job name is specified with the **-J** option, the **bjobs**, **bhist**, and **bacct** commands display the command as the job name.

The commands are run in the order in which they are given.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

bsub, **tspeek**

Chapter 108. wgpasswd

Changes a user's password for a Microsoft Windows workgroup.

Synopsis

`wgpasswd` [*user_name*]

`wgpasswd` [`-h`]

Description

Important: You must run this command on a host in a Microsoft Windows workgroup. You must have administrative privileges to change another user's password.

Prompts for old and new passwords, then changes the password on every host in the workgroup.

By default, modifies your own user account.

Options

user_name
Specifies the account to modify. You must have administrative privileges to change another user's password.

-h
Prints command usage to `stderr` and exits.

Output

For each host in the workgroup, returns the status of the operation (`SUCCESS` or `FAILED`).

Files

Modifies the LSF password file.

Chapter 109. wguser

Modifies user accounts for a Microsoft Windows workgroup

Synopsis

`wguser` [`-r`] *user_name* ...

`wguser` [`-h`]

Description

Important: You must run this command on a host in a Microsoft Windows workgroup. You must have administrative privileges on every host in the workgroup.

Modifies accounts on every host in the workgroup that you have administrative privileges on.

By default, prompts for a default password to use for all of the accounts, and then creates the specified user accounts on each host, if they do not exist.

Use `-r` to remove accounts from the workgroup.

Options

-r
Removes the specified user accounts from each host, if they exist.

***user_name* ...**
Required. Specifies the accounts to add or remove.

-h
Prints command usage to `stderr` and exits.

Output

For each host in the workgroup, returns the result of the operation (`SUCCESS` or `FAILED`).