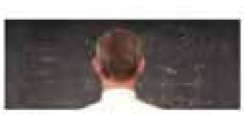
Platform **LSF**® Reference

Version 6.2
March 2006
Comments to: doc@platform.com



We'd like to hear from you

You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this document, please address your comments to doc@platform.com.

Your comments should pertain only to Platform documentation. For product support, contact support@platform.com.

Contents

Welcome 7

Part I: Commands

bacct 13
badmin 24
bbot 35
bchkpnt 37
bclusters 39
bgadd 41
bgdel 42
bhist 43
bhosts 49
bhpart 56
bjgroup 58
bjobs 60
bkill 70
bladmin 75
blcollect 76
blhosts 78
blimits 79
blinfo 83
blkill 87
blstat 88
bltasks 92
blusers 95
bmgroup 98
bmig 99

bmod . . . 101
bparams . . . 106
bpeek . . . 107
bpost . . . 108
bqueues . . . 110
bread . . . 123
brequeue . . . 125
bresources . . . 127
brestart . . . 128
bresume . . . 130
brlainfo . . . 132
brsvadd . . . 134
brsvdel . . . 138
brsvs . . . 139
brun . . . 141
bsla . . . 143
bstatus . . . 146
bstop . . . 148
bsub . . . 150
bswitch . . . 176
btop . . . 178
bugroup . . . 180
busers . . . 181
ch . . . 183
lsacct . . . 186
lsacctmrg . . . 189
lsadmin . . . 190
lsclusters . . . 198
lseligible . . . 200
lsfinstall . . . 202
lsfmon . . . 208
lsfrestart . . . 209

lsfsetcluster . . . 210
lsfshutdown . . . 211
lsfstartup . . . 212
lsgrun . . . 213
lshosts . . . 216
lsid . . . 220
lsinfo . . . 221
lsload . . . 223
lsloadadj . . . 228
lslogin . . . 230
lsltasks . . . 232
lsmake . . . 234
lsmon . . . 236
lspasswd . . . 240
lsplace . . . 241
lsrcp . . . 243
lsrtasks . . . 246
lsrun . . . 248
lstcsh . . . 251
pam . . . 256
taskman . . . 260
wgpasswd . . . 261
wguser . . . 263

Part II: Environment Variables

Environment Variables . . . 267

Part III: Configuration Files

bld.license.acct . . . 295
cshrc.lsf and profile.lsf . . . 297
hosts . . . 303
install.config . . . 307
lim.acct . . . 313
lsb.acct . . . 315
lsb.events . . . 323
lsb.hosts . . . 353
lsb.modules . . . 367
lsb.params . . . 373
lsb.queues . . . 399
lsb.resources . . . 433
lsb.serviceclasses . . . 457
lsb.users . . . 465
lsf.acct . . . 475
lsf.cluster . . . 479
lsf.*cluster_name*.license.acct . . . 499
lsf.conf . . . 503
lsf.licensescheduler . . . 577
lsf.shared . . . 599
lsf.sudoers . . . 607
lsf.task . . . 617
setup.config . . . 623
slave.config . . . 627
win_install.config . . . 633

Part IV: Troubleshooting

Troubleshooting and Error Messages . . . 641
Index . . . 653

Welcome

Contents

- "About This Guide" on page 8
- "Learn About Platform Products" on page 9
- "Get Technical Support" on page 10

About Platform Computing

Platform Computing is the largest independent grid software developer, delivering intelligent, practical enterprise grid software and services that allow organizations to plan, build, run and manage grids by optimizing IT resources. Through our proven process and methodology, we link IT to core business objectives, and help our customers improve service levels, reduce costs and improve business performance.

With industry-leading partnerships and a strong commitment to standards, we are at the forefront of grid software development, propelling over 1,600 clients toward powerful insights that create real, tangible business value. Recognized worldwide for our grid computing expertise, Platform has the industry's most comprehensive desktop-to-supercomputer grid software solutions. For more than a decade, the world's largest and most innovative companies have trusted Platform Computing's unique blend of technology and people to plan, build, run and manage grids.

Learn more at www.platform.com.

About This Guide

Last update March 1 2006

Latest version www.platform.com/Support/Documentation.htm

Purpose of this guide

This guide provides reference information for the Platform LSF® software ("LSF"). It covers the following topics:

- LSF commands
- Environment variables
- Configuration files
- Troubleshooting

Who should use this guide

This guide accompanies *Administering Platform LSF*, and is your source for reference information.

Typographical conventions

Typeface	Meaning	Example
`Courier`	The names of on-screen computer output, commands, files, and directories	The `lsid` command
`Bold Courier`	What you type, exactly as shown	Type **`cd /bin`**
Italics	◆ Book titles, new words or terms, or words to be emphasized ◆ Command-line place holders—replace with a real name or value	The queue specified by *queue_name*
Bold Sans Serif	◆ Names of GUI elements that you manipulate	Click **OK**

Command notation

Notation	Meaning	Example
Quotes " or '	Must be entered exactly as shown	"*job_ID*[*index_list*]"
Commas ,	Must be entered exactly as shown	-C *time0*,*time1*
Ellipsis …	The argument before the ellipsis can be repeated. Do not enter the ellipsis.	*job_ID* …
lower case italics	The argument must be replaced with a real value you provide.	*job_ID*
OR bar |	You must enter one of the items separated by the bar. You cannot enter more than one item, Do not enter the bar.	[-h | -V]
Parenthesis ( )	Must be entered exactly as shown	-X "*exception_cond*([*params*])::*action*] ...
Option or variable in square brackets [ ]	The argument within the brackets is optional. Do not enter the brackets.	lsid [-h]
Shell prompts	◆ C shell: `%` ◆ Bourne shell and Korn shell: `$` ◆ root account: `#` Unless otherwise noted, the C shell prompt is used in all command examples	`%` **`cd /bin`**

Learn About Platform Products

World Wide Web and FTP

The latest information about all supported releases of Platform LSF is available on the Platform Web site at www.platform.com. Look in the Online Support area for current Release Notes, Upgrade Notices, Frequently Asked Questions (FAQs), Troubleshooting, and other helpful information.

The Platform FTP site (ftp.platform.com) also provides current Release Notes, and Upgrade information for all supported releases of Platform LSF.

Visit the Platform User Forum at www.platformusers.net to discuss workload management and strategies pertaining to distributed and Grid Computing.

If you have problems accessing the Platform web site or the Platform FTP site, contact support@platform.com.

Platform training

Platform's Professional Services training courses can help you gain the skills necessary to effectively install, configure and manage your Platform products. Courses are available for both new and experienced users and administrators at our corporate headquarters and Platform locations worldwide.

Customized on-site course delivery is also available.

Find out more about Platform Training at www.platform.com/training, or contact Training@platform.com for details.

Release notes and UPGRADE

Before installing LSF, be sure to read the file named `release_notes.html`. To upgrade to Version 6.2, follow the steps in `upgrade.html`.

Platform documentation

Documentation for Platform products is available in HTML and PDF format on the Platform Web site at www.platform.com/Support/Documentation.htm.

Get Technical Support

Contact Platform

Contact Platform Computing or your LSF vendor for technical support. Use one of the following to contact Platform technical support:

Email support@platform.com

World Wide Web www.platform.com

Mail Platform Support
Platform Computing Corporation
3760 14th Avenue
Markham, Ontario
Canada L3R 3T7

When contacting Platform, please include the full name of your company.

See the Platform Web site at www.platform.com/contactus for other contact information.

Get patch updates and other notifications

To get periodic patch update information, critical bug notification, and general support notification from Platform Support, contact supportnotice-request@platform.com with the subject line containing the word "subscribe".

To get security related issue notification from Platform Support, contact securenotice-request@platform.com with the subject line containing the word "subscribe".

We'd like to hear from you

If you find an error in any Platform documentation, or you have a suggestion for improving it, please let us know:

Email doc@platform.com

Mail Information Development
Platform Computing Corporation
3760 14th Avenue
Markham, Ontario
Canada L3R 3T7

Be sure to tell us:

- The title of the manual you are commenting on
- The version of the product you are using
- The format of the manual (HTML or PDF)

PART I

Commands

bacct

displays accounting statistics about finished jobs

SYNOPSIS

bacct [**-b** | **-l**] [**-d**] [**-e**] [**-w**] [**-C** *time0*,*time1*] [**-D** *time0*,*time1*] [**-f** *logfile_name*]
[**-Lp** *ls_project_name* ...] [**-m** *host_name* ...]
[**-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**] [**-x**] [*job_ID* ...]

bacct -U *reservation_ID* ... | **-U all** [**-u** *user_name* ... | **-u all**]

bacct [**-h** | **-V**]

DESCRIPTION

By default, displays accounting statistics for all finished jobs (with a DONE or EXIT status) submitted by the user who invoked the command, on all hosts, projects, and queues in the LSF system.

By default, `bacct` displays statistics for all jobs logged in the current LSF accounting log file: `LSB_SHAREDIR/`*`cluster_name`*`/logdir/lsb.acct` (see `lsb.acct`(5)).

By default, CPU time is not normalized.

If neither `-l` nor `-b` is present, displays the fields in SUMMARY only (see OUTPUT).

Statistics not reported by `bacct` but of interest to individual system administrators can be generated by directly using `awk`(1) or `perl`(1) to process the `lsb.acct` file.

All times are in seconds.

When combined with the `-U` option, `-u` is interpreted as the user name of the reservation creator. For example:

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

Shows all the advance reservations created by user `user2`.

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

In a MultiCluster 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.

Throughput Calculation

The throughput (T) of the LSF system, certain hosts, or certain queues is calculated by the formula:

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

where:

- N is the total number of jobs for which accounting statistics are reported
- BT is the Start time—when the first job was logged
- ET is the End time—when the last job was logged

You can use the option `-C` *time0,time1* to specify the Start time as *time0* and the End time as *time1*. In this way, you can examine throughput during a specific time period.

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

The total throughput of the LSF system can be calculated by specifying `-u all` without any of the `-m`, `-q`, `-S`, `-D` or *job_ID* options. The throughput of certain hosts can be calculated by specifying `-u all` without the `-q`, `-S`, `-D` or *job_ID* options. The throughput of certain queues can be calculated by specifying `-u all` without the `-m`, `-S`, `-D` or *job_ID* options.

`bacct` does not show local pending batch jobs killed using `bkill -b`. `bacct` shows MultiCluster jobs and local running jobs even if they are killed using `bkill -b`.

OPTIONS

-b

Brief format. Displays accounting statistics in brief format. See "OUTPUT" for a description of information that is displayed.

-d

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

-e

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

-l

Long format. Displays additional accounting statistics. See "OUTPUT" for a description of information that is displayed.

-w

Wide format. Displays accounting statistics in a wide format without truncating the fields.

-C *time0,time1*

Displays accounting statistics for jobs that completed or exited during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist`(1).

-D *time0,time1*

Displays accounting statistics for jobs dispatched during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist`(1).

-f *logfile_name*

Searches the specified job log file for accounting statistics. Specify either an absolute or relative path.

Useful for offline analysis.

-Lp *ls_project_name ...*

Displays accounting statistics for jobs belonging to the specified 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_name ...*

Displays accounting statistics for jobs 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 (').

-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 `bacct` offline by indicating a job log file, you must specify a CPU factor.

Use `lsinfo` to get host model and CPU factor information.

-P *project_name ...*

Displays accounting statistics for jobs belonging to 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 accounting statistics for jobs 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 submitted during the specified time interval. Reads `lsb.acct` and all archived log files (`lsb.acct.`*n*) unless `-f` is also used.

The time format is the same as in `bhist`(1).

-sla *service_class_name*

Displays accounting statistics for jobs that ran under the specified service class.

Use `bsla` to display the properties of service classes configured in `LSB_CONFDIR/`*cluster_name*`/configdir/lsb.serviceclasses` (see `lsb.serviceclasses`(5)) and dynamic information about the state of each 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 (').

In a MultiCluster environment, you cannot see information about remote reservations. You cannot specify a remote reservation ID, and the keyword `all` only displays information about reservations in the local cluster.

-u *user_name* **...|-u all**

Displays accounting statistics for jobs 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.

-x

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

job_ID ...

Displays accounting statistics for jobs with the specified job IDs.

This option overrides all other options except `-b`, `-l`, `-f`, `-h`, and -V. If the reserved job ID 0 is used, it will be ignored.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

SUMMARY (default format)

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
- Total throughput
- Beginning time: the completion or exit time of the first job selected
- Ending time: 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 consumed by a job divided by its turnaround time.

The throughput is the number of completed jobs divided by the time period to finish these jobs (jobs/hour). For more details, see "DESCRIPTION" on page 13.

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`(3), 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 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

Name of the job (see `bsub`(1)).

Long Format (-l)

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

JOBID

Identifier that LSF assigned to the job.

PROJECT_NAME

Project name assigned to the job.

STATUS

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

DISPAT_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, in kilobytes.

SWAP

Maximum virtual memory usage of all processes in a job, in kilobytes.

CWD

Current working directory of the job.

INPUT_FILE

File from which the job reads its standard input (see `bsub`(1)).

OUTPUT_FILE

File to which the job writes its standard output (see `bsub`(1)).

ERR_FILE

File in which the job stores its standard error output (see `bsub`(1)).

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.

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 `bsub -U`

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 separated by slashes (`month/day/hour/minute`). For example:

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

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

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

Termination reasons displayed by bacct

When LSF detects that a job is terminated, `bacct -l` displays one of the following termination reasons:

- TERM_ADMIN: Job killed by root or LSF administrator
- TERM_CHKPNT: Job killed after checkpointing
- TERM_CPULIMIT: Job killed after reaching LSF CPU usage limit
- TERM_DEADLINE: Job killed after deadline expires
- TERM_EXTERNAL_SIGNAL: Job killed by a signal external to LSF
- TERM_FORCE_ADMIN: Job killed by root or LSF administrator without time for cleanup
- TERM_FORCE_OWNER: Job killed by owner without time for cleanup
- TERM_LOAD: Job killed after load exceeds threshold
- TERM_MEMLIMIT: Job killed after reaching LSF memory usage limit
- TERM_OWNER: Job killed by owner
- TERM_PREEMPT: Job killed after preemption
- TERM_PROCESSLIMIT: Job killed after reaching LSF process limit
- TERM_REQUEUE_ADMIN: Job killed and requeued by root or LSF administrator
- TERM_REQUEUE_OWNER: Job killed and requeued by owner
- TERM_RUNLIMIT: Job killed after reaching LSF run time limit
- TERM_SLURM: Job terminated abnormally in SLURM (node failure)
- TERM_SWAP: Job killed after reaching LSF swap usage limit
- TERM_THREADLIMIT: Job killed after reaching LSF thread limit
- TERM_WINDOW: Job killed after queue run window closed
- TERM_ZOMBIE: Job exited while LSF is not available

See lsbatch.h for the mapping between the integer value logged to lsb.acct and termination reason keyword.

EXAMPLES

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:      60      Total number of exited jobs:   118
 Total CPU time consumed:    1011.5      Average CPU time consumed:     5.7
 Maximum CPU time of a job:   991.4      Minimum CPU time of a job:     0.0
 Total wait time in queues: 134598.0
 Average wait time in queue:  756.2
 Maximum wait time in queue: 7069.0      Minimum wait time in queue:    0.0
 Average turnaround time:      3585 (seconds/job)
 Maximum turnaround time:     77524      Minimum turnaround time:         6
 Average hog factor of a job:  0.00 ( cpu time / turnaround time )
 Maximum hog factor of a job:  0.56      Minimum hog factor of a job:  0.00
 Total throughput:             0.67 (jobs/hour)  during  266.18 hours
 Beginning time:       Aug  8 15:48      Ending time:          Aug 19 17:59
```

Jobs that have 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: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File
                     </dev/null>;
Mon Aug 11 18:17:22: Dispatched to <hostC>;
Mon Aug 11 18:18:54: 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>
Tue Aug 12 14:15:03: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File
                     </dev/null>;
Tue Aug 12 14:15:15: Dispatched to <hostC>;
Tue Aug 12 14:25:08: 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: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File
                     </dev/null>;
Tue Aug 12 14:26:18: Dispatched to <hostC>;
Tue Aug 12 14:33:16: 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>
Mon Aug 18 20:27:44: Submitted from host <hostB>, CWD <$HOME/jobs>, Output File
                     </dev/null>;
Mon Aug 18 20:31:16: [14] dispatched to <hostA>;
Mon Aug 18 20:31:18: 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
 Total throughput:             0.59 (jobs/hour)  during  170.21 hours
 Beginning time:       Aug 11 18:18      Ending time:          Aug 18 20:31
```

Advance reservation accounting information

```
% bacct -U user1#2
Accounting for:
  - advanced reservation IDs: user1#2
  - advanced 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
```

LSF Job termination reason logging

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

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

Use `bacct -l` to view job exit information logged to `lsb.acct`:

```
% 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>
Thu Sep 16 15:22:09: Submitted from host <hostA>, CWD <$HOME>;
Thu Sep 16 15:22:20: Dispatched to 4 Hosts/Processors <4*hostA>;
Thu Sep 16 15:22:20: slurm_id=21793;ncpus=4;slurm_alloc=n[13-14];
```

```
Thu Sep 16 15:23:21: 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
```

FILES

Reads `lsb.acct`, `lsb.acct.`*n*.

SEE ALSO

`bhist`(1), `bsub`(1), `bjobs`(1), `lsb.acct`(5), `brsvadd`(8), `brsvs`(1), `bsla`(1), `lsb.serviceclasses`(5)

badmin

administrative tool for LSF

SYNOPSIS

badmin *subcommand*

badmin [**-h** | **-V**]

SUBCOMMAND LIST

ckconfig [**-v**]
diagnose [*job_ID* ... | **"***job_ID***[***index***]"** ...]
reconfig [**-v**] [**-f**]
mbdrestart [**-C** *comment*] [**-v**] [**-f**]
qopen [**-C** *comment*] [*queue_name* ... | **all**]
qclose [**-C** *comment*] [*queue_name* ... | **all**]
qact [**-C** *comment*] [*queue_name* ... | **all**]
qinact [**-C** *comment*] [*queue_name* ... | **all**]
qhist [**-t** *time0,time1*] [**-f** *logfile_name*] [*queue_name* ...]
hopen [**-C** *comment*] [*host_name* ... | *host_group* ... | **all**]
hclose [**-C** *comment*] [*host_name* ... | *host_group* ... | **all**]
hrestart [**-f**] [*host_name* ... | **all**]
hshutdown [**-f**] [*host_name* ... | **all**]
hstartup [**-f**] [*host_name* ... | **all**]
hhist [**-t** *time0,time1*] [**-f** *logfile_name*] [*host_name* ...]
mbdhist [**-t** *time0,time1*] [**-f** *logfile_name*]
hist [**-t** *time0,time1*] [**-f** *logfile_name*]
hghostadd [**-C** *comment*] *host_group host_name* [*host_name* ...]
hghostdel [**-f**] [**-C** *comment*] *host_group host_name* [*host_name* ...]
help [*command* ...] | **?** [*command* ...]
quit
mbddebug [**-c** *class_name* ...] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**]
mbdtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**]
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**]
schdtime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**]
-h
-V

DESCRIPTION

This command can only be used by LSF administrators.

badmin provides a set of commands to control and monitor LSF. If no subcommands are supplied for badmin, badmin prompts for a command from standard input.

Information about each command is available through the help command.

The `badmin` commands consist of a set of privileged commands and a set of non-privileged commands. Privileged commands can only be invoked by root or LSF administrators as defined in the configuration file (see `lsf.cluster.cluster`(5) for `ClusterAdmin`). Privileged commands are:

`reconfig`

`mbdrestart`

`qopen`

`qclose`

`qact`

`qinact`

`hopen`

`hclose`

`hrestart`

`hshutdown`

`hstartup`

`diagnose`

The configuration file `lsf.sudoers`(5) has to be set in order to use the privileged command `hstartup` by a non-root user.

All other commands are non-privileged commands and can be invoked by any LSF user. If the privileged commands are to be executed by the LSF administrator, `badmin` must be installed `setuid root`, because it needs to send the request using a privileged port.

For subcommands for which multiple hosts can be specified, do not enclose the host names in quotation marks.

Obsolete commands

Commands `bqc`(8), `breconfig`(8) and `breboot`(8) are superseded by `badmin`(8).

OPTIONS

`subcommand`

Executes the specified subcommand. See 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 located in the `LSB_CONFDIR/cluster_name/configdir` directory. Also checks configuration in `LSF_ENVDIR/lsf.licensescheduler`.

The LSB_CONFDIR variable is defined in `lsf.conf` (see `lsf.conf`(5)) which is in LSF_ENVDIR or `/etc` (if LSF_ENVDIR is not defined).

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

-v

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

diagnose [*job_ID* ... | **"***job_ID***[***index***]"** ...]

Displays full pending reason list if CONDENSE_PENDING_REASONS=Y is set in `lsb.params`. For example:

```
% badmin diagnose 1057
```

reconfig [**-v**] [**-f**]

Dynamically reconfigures LSF without restarting `mbatchd`.

Configuration files are checked for errors and the results displayed to `stderr`. If no errors are found in the configuration files, a reconfiguration request is sent to `mbatchd` and configuration files are reloaded.

With this option, `mbatchd` and `mbschd` are not restarted and `lsb.events` is not replayed. To restart `mbatchd` and `mbschd`, and replay `lsb.events`, use `badmin mbdrestart`.

When you issue this command, `mbatchd` 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, `badmin` prompts you to display detailed messages. If fatal errors are found, reconfiguration is not performed, and `badmin` exits.

If you add a host to a queue, the new host will not be 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`.

If you add a host to a host group, the new host will not be 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`.

-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 fatal errors.

mbdrestart [**-C** *comment*] [**-v**] [**-f**]

Dynamically reconfigures LSF and restarts `mbatchd` and `mbschd`.

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

If warning errors are found, `badmin` prompts you to display detailed messages. If fatal errors are found, `mbatchd` and `mbschd` restart is not performed, and `badmin` exits.

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

-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.

-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` restart to proceed if configuration files contain no fatal errors.

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

Opens specified queues, or all queues if the reserved word `all` is specified. If no queue is specified, the system default queue is assumed (see `lsb.queues`(5) for DEFAULT_QUEUE). A queue can accept batch 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.

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

Closes specified queues, or all queues if the reserved word `all` is specified. If no queue is specified, the system default queue is assumed. A queue will not accept any job if it is closed.

-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.

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

Activates specified queues, or all queues if the reserved word `all` is specified. If no queue is specified, the system default queue is assumed. Jobs in a queue can be dispatched if the queue is activated.

A queue inactivated by its run windows cannot be reactivated by this command (see `lsb.queues(5)` for RUN_WINDOW).

-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.

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

Inactivates specified queues, or all queues if the reserved word `all` is specified. If no queue is specified, the system default queue is assumed. No job in a queue can be dispatched if the queue is inactivated.

-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.

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 `bhist(1)` for the time format. The default is to display all queue events in the event log file (see below).

-f *logfile_name*

Specify the file name of the event log file. Either an absolute or a relative path name may be specified. The default is to use the event log file currently used by 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 commands `qclose`, `qopen`, `qact`, and `qinact`, `qhist` displays the comment text.

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

Opens batch server hosts. Specify the names of any server hosts or host groups (see `bmgroup(1)`). All batch server hosts will be opened if the reserved word `all` is specified. If no host or host group is specified, the local host is assumed. A host accepts batch jobs 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. If you open a host group, each host group member displays with the same comment string.

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

Closes batch server hosts. Specify the names of any server hosts or host groups (see `bmgroup(1)`). All batch server hosts will be closed if the reserved word `all` is specified. If no argument is specified, the local host is assumed. A closed host will not accept any new job, but jobs already dispatched to the host will not be affected. Note that this is different from a host closed by a window; all jobs on it are suspended in that case.

-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. If you close a host group, each host group member displays with the same comment string.

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

Restarts `sbatchd` 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. `sbatchd` will rerun itself from the beginning. This allows new `sbatchd` binaries to be used.

-f

Disables interaction and does not ask for confirmation for restarting `sbatchd`.

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

Shuts down `sbatchd` 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. `sbatchd` will exit upon receiving the request.

-f

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

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

Starts `sbatchd` on the specified hosts, or on all batch server hosts if the reserved word `all` is specified. Only `root` and users listed in the file `lsf.sudoers(5)` can use the `all` and `-f` options. These users must be able to use `rsh` or `ssh` on all LSF hosts without having to type in passwords. If no host is specified, the local host is assumed.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

-f

Disables interaction and does not ask for confirmation for starting `sbatchd`.

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. Options `-t` and `-f` are exactly the same as those of `qhist` (see above).

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

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

Displays historical events for `mbatchd`. Events describe the starting and exiting of `mbatchd`. Options `-t` and `-f` are exactly the same as those of `qhist` (see above).

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

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

Displays historical events for all the queues, hosts and `mbatchd`. Options `-t` and `-f` are exactly the same as those of `qhist` (see above).

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

hghostadd [**-C** *comment*] *host_group host_name* [*host_name* ...]

Dynamically adds hosts to a host group. After receiving the host information from the master LIM, `mbatchd` dynamically adds the host without triggering a `reconfig`.

Once the host is added to the group, it will be considered to be part of that group with respect to scheduling decision making for both newly submitted jobs and for existing pending jobs.

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

This command also fails if you try to add dynamic hosts to condensed host groups.

To enable dynamic host configuration, define `LSF_MASTER_LIST` and `LSF_DYNAMIC_HOST_WAIT_TIME` in `lsf.conf` and `LSF_HOST_ADDR_RANGE` in `lsf.cluster.`*cluster_name*.

-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.

hghostdel [**-f**] [**-C** *comment*] *host_group host_name* [*host_name* ...]

Dynamically deletes hosts from a host group by triggering an `mbatchd reconfig`

The host must be dynamic, otherwise it will not be deleted from a host group that is defined in the `lsb.hosts` file. This command fails if any of the specified host groups or host names are not valid.

This command fails if you try to delete dynamic hosts from condensed host groups.

To enable dynamic host configuration, define `LSF_MASTER_LIST` and `LSF_DYNAMIC_HOST_WAIT_TIME` in `lsf.conf` and `LSF_HOST_ADDR_RANGE` in `lsf.cluster.`*cluster_name*.

When a dynamic host is configured as a static host in `lsf.cluster.`*cluster_name*, run hghostdel to remove the host from the host group as a dynamic member.

-f

Disables interaction and does not prompt for confirmation before forcing an `mbdreconfig`.

-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.

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

Displays the syntax and functionality of the specified commands.

quit

Exits the `badmin` session.

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

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

See `sbddebug` for an explanation of options.

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

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

See `sbdtime` for an explanation of options.

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

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

In MultiCluster, debug levels can only be set for hosts within the same cluster. For example, you could not 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=current 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 command was submitted)

-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 `lsf.h`.

Possible classes:

LC_AFS - Log AFS messages

LC_AUTH - Log authentication messages

LC_CHKPNT - Log checkpointing messages

LC_COMM - Log communication messages

LC_DCE - Log messages pertaining to DCE support

LC_EEVENTD - Log `eeventd` messages

LC_EXEC - Log significant steps for job execution

LC_FAIR - Log fairshare policy messages

LC_FILE - Log file transfer messages

LC_HANG - Mark where a program might hang

LC_JLIMIT - Log job slot limit messages

LC_LICENCE - Log license management messages

LC_LOADINDX - Log load index messages

LC_M_LOG - Log multievent logging messages

LC_MPI - Log MPI messages

LC_MULTI - Log messages pertaining to MultiCluster

LC_PEND - Log messages related to job pending reasons

LC_PERFM - Log performance messages

LC_PIM - Log PIM messages

LC_PREEMPT - Log preemption policy messages

LC_SIGNAL - Log messages pertaining to signals

LC_SYS - Log system call messages

LC_TRACE - Log significant program walk steps

LC_XDR - Log everything transferred by XDR

Default: 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 levels.

Possible values:

`0` LOG_DEBUG level in parameter LSF_LOG_MASK in `lsf.conf`.

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

`2` LOG_DEBUG2 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 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.

Default: 0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

-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 may be specified.

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

The name of the file that will be created will have 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.

Default: 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 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.

Lists of host names must be separated by spaces and enclosed in quotation marks.

Default: local host (host from which command was submitted)

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

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

In MultiCluster, timing levels can only be set for hosts within the same cluster. For example, you could not 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 host (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.

Valid values: 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.

Default: undefined (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 may be specified.

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

The name of the file created 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.

Default: 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 starting 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.

Lists of hosts must be separated by spaces and enclosed in quotation marks.

Default: local host (host from which command was submitted)

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

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

See `sbddebug` for an explanation of options.

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

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

See `sbdtime` for an explanation of options.

SEE ALSO

`bqueues`(1), `bhosts`(1), `lsb.params`(5), `lsb.queues`(5), `lsb.hosts`(5), `lsf.conf`(5), `lsf.cluster`(5), `sbatchd`(8), `mbatchd`(8), `mbschd`(8)

bbot

moves a pending job relative to the last job in the queue

SYNOPSIS

bbot *job_ID* | **"***job_ID***[***index_list***]"** [*position*]

bbot [**-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 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 only operate on their own jobs, whereas the LSF administrator can operate on any user's jobs. Users can only change the relative position of their own jobs.

If invoked by the LSF administrator, `bbot` moves the selected job after the last job with the same priority submitted to the queue. The positions of all users' jobs in the queue can be changed by the LSF administrator.

If invoked by a regular user, `bbot` moves the selected job after the last job with the same priority submitted by the user to the queue.

Pending jobs are displayed by `bjobs` in the order in which they will be considered for dispatch.

A user may use `bbot` to change the dispatch order of their jobs scheduled using a fairshare policy. However, if a job scheduled using a fairshare policy is moved by the LSF administrator using `btop`, the job will not be subject to further fairshare scheduling unless the same job is subsequently moved by the LSF administrator using `bbot`; in this case the job will be scheduled again using the same fairshare policy (see the `FAIRSHARE` keyword in `lsb.queues`(5) and `HostPartition` keyword in `lsb.hosts`(5)).

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

OPTIONS

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 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 share 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. *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`(1), `bswitch`(1), `btop`(1), JOB_POSITION_CONTROL_BY_ADMIN in `lsb.params`

bchkpnt

checkpoints one or more checkpointable jobs

SYNOPSIS

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

bchkpnt [**-f**] [**-k**] [**-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 your running (RUN) or suspended (SSUSP, USUSP, and PSUSP) checkpointable jobs. LSF administrators and root can checkpoint jobs submitted by other users.

By default, checkpoints one job, the most recently submitted job, or the most recently submitted job that also satisfies other specified options (`-m`, `-q`, `-u` and `-J`). Specify `-0` (zero) to checkpoint multiple jobs. Specify a job ID to checkpoint one specific job.

By default, jobs continue to execute after they have been checkpointed.

To submit a checkpointable job, use `bsub -k` or submit the job to a checkpoint queue (CHKPNT in `lsb.queues`(5)). Use `brestart(1)` to start checkpointed jobs.

LSF invokes the `echkpnt`(8) executable found in LSF_SERVERDIR to perform the checkpoint.

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

OPTIONS

0

(Zero). Checkpoints multiple jobs. Checkpoints all the jobs that satisfy other specified options (`-m`, `-q`, `-u` and `-J`).

-f

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

-k

Kills a job after it has been 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. To allow your job to make progress while still providing fault tolerance, specify a checkpoint period of 30 minutes or longer.

-J *job_name*

Only checkpoints jobs that have the specified job name.

-m *host_name* | **-m** *host_group*

Only checkpoints jobs dispatched to the specified hosts.

-q *queue_name*

Only checkpoints jobs dispatched from the specified queue.

-u "*user_name***"** | **-u all**

Only checkpoints jobs submitted by the specified users. The keyword `all` specifies all users. Ignored if a job ID other than 0 (zero) is specified.

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 issued by root or the LSF administrator, will checkpoint and kill all checkpointable jobs on `hostA`. This is useful when a host needs to be shut down or rebooted.

SEE ALSO

`bsub`(1), `bmod`(1), `brestart`(1), `bjobs`(1), `bqueues`(1), `bhosts`(1), `libckpt.a`(3), `lsb.queues`(5), `echkpnt`(8), `erestart`(8), `mbatchd`(8)

bclusters

displays status of MultiCluster connections

SYNOPSIS

bclusters [**-h** | **-V**]

DESCRIPTION

Displays a list of MultiCluster queues together with their relationship with queues in remote clusters.

OPTIONS

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

Job Forwarding Model

Information related to the job forwarding model is displayed under the heading Remote Batch Information.

LOCAL_QUEUE

Name of a local MultiCluster send-jobs or receive-jobs queue.

JOB_FLOW

Indicates direction of job flow.

send

The local queue is a MultiCluster send-jobs queue (SNDJOBS_TO is defined in the local queue).

recv

The local queue is a MultiCluster 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 “-”.

CLUSTER

For send-jobs queues, shows the name of the remote cluster containing 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 has not been established. This could occur because there are no jobs waiting to be dispatched, or because the remote master cannot be located.

Resource Leasing Model

Information 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

MultiCluster jobs can run.

disc

No communication between the two clusters. This could be a temporary situation or could indicate a MultiCluster configuration error.

conn

The two clusters communicate, but the lease is not established. This should be a temporary situation.

FILES

Reads `lsb.queues`.

SEE ALSO

`bhosts`(1) displays detailed information about leased resources.

`bqueues`(1) displays information about local MultiCluster queues.

`lsclusters`(1), `ls_info`(3), `ls_policy`(3), `lsb.queues`(5)

bgadd

creates job groups

SYNOPSIS

bgadd *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

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

- `% bgadd /risk_group`
  creates a job group named `risk_group` under the root group `/`.
- `% bgadd /risk_group/portfolio1`
  creates a job group named `portfolio1` under job group `/risk_group`.

SEE ALSO

`bgdel`(1), `bjgroup`(1)

bgdel

deletes job groups

SYNOPSIS

bgdel *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. The job group cannot contain any jobs.

OPTIONS

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

```
% bgdel /risk_group
Job group /risk_group is deleted.
```

deletes the job group `/risk_group` and all its subgroups.

SEE ALSO

`bgadd`(1), `bjgroup`(1)

bhist

displays historical information about jobs

SYNOPSIS

bhist [**-a** | **-d** | **-p** | **-r** | **-s**] [**-b** | **-w**] [**-l**] [**-t**] [**-C** *start_time*,*end_time*]
[**-D** *start_time*,*end_time*] [**-S** *start_time*,*end_time*] [**-T** *start_time*,*end_time*]
[**-f** *logfile_name* | **-n** *number_logfiles* | **-n 0**] [**-J** *job_name*]
[**-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**]

bhist [**-J** *job_name*] [**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*]
[*job_ID* ... | **"***job_ID***[***index***]"** ...]

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
- Searches the event log file currently used by the LSF system: `$LSB_SHAREDIR/`*`cluster_name`*`/logdir/lsb.events` (see `lsb.events`(5))
- Displays events occurring in the past week, but this can be changed by setting the environment variable LSB_BHIST_HOURS to an alternative number of hours

If neither `-l` nor `-b` is present, the default is to display the fields in "OUTPUT" only.

OPTIONS

-a

Displays information about both finished and unfinished jobs.

This option overrides `-d`, `-p`, `-s`, and `-r`.

-b

Brief format. Displays the information in a brief format. If used with the `-s` option, shows the reason why each job was suspended.

-d

Only displays information about finished jobs.

-l

Long format. Displays additional information. If used with `-s`, shows the reason why each job was suspended.

If you submitted a job using the `OR` (||) expression to specify alternative resources, this option displays the successful `rusage` string that caused the job to run.

`bhist -l` can display job exit codes. A job with exit code 131 means that the job exceeded a configured resource usage limit and LSF killed the job with signal 3 (131-128=3).

-p

Only displays information about pending jobs.

-r

Only displays information about running jobs.

-s

Only displays information about suspended jobs.

-t

Displays job events chronologically.

-w

Wide format. Displays the information in a wide format.

-C *start_time,end_time*

Only displays jobs that completed or exited during the specified time interval. Specify the span of time for which you want to display the history. If you do not specify a start time, the start time is assumed to be the time of the first occurrence. If you do not specify an end time, the end time is assumed to be now. If you do not specify an end time, the end time is assumed to be now.

Specify the times in the format "*yyyy*/*mm*/*dd*/*HH*:*MM*". Do not specify spaces in the time interval string.

The time interval can be specified in many ways. For more specific syntax and examples of time formats, see TIME INTERVAL FORMAT.

-D *start_time,end_time*

Only displays jobs dispatched during the specified time interval. Specify the span of time for which you want to display the history. If you do not specify a start time, the start time is assumed to be the time of the first occurrence. If you do not specify an end time, the end time is assumed to be now. If you do not specify an end time, the end time is assumed to be now.

Specify the times in the format "*yyyy*/*mm*/*dd*/*HH*:*MM*". Do not specify spaces in the time interval string.

The time interval can be specified in many ways. For more specific syntax and examples of time formats, see TIME INTERVAL FORMAT.

-S *start_time,end_time*

Only displays information about jobs submitted during the specified time interval. Specify the span of time for which you want to display the history. If you do not specify a start time, the start time is assumed to be the time of the first occurrence. If you do not specify an end time, the end time is assumed to be now. If you do not specify an end time, the end time is assumed to be now.

Specify the times in the format "*yyyy*/*mm*/*dd*/*HH*:*MM*". Do not specify spaces in the time interval string.

The time interval can be specified in many ways. For more specific syntax and examples of time formats, see TIME INTERVAL FORMAT.

-T *start_time,end_time*

Used together with `-t`.

Only displays information about job events within the specified time interval. Specify the span of time for which you want to display the history. If you do not specify a start time, the start time is assumed to be the time of the first occurrence. If you do not specify an end time, the end time is assumed to be now. If you do not specify an end time, the end time is assumed to be now.

Specify the times in the format "*yyyy*/*mm*/*dd*/*HH*:*MM*". Do not specify spaces in the time interval string.

The time interval can be specified in many ways. For more specific syntax and examples of time formats, see TIME INTERVAL FORMAT.

-f *logfile_name*

Searches the specified event log. Specify either an absolute or a relative path.

Useful for analysis directly on the file.

-J *job_name*

Only displays the jobs that have the specified *job_name*.

-Lp *ls_project_name*

Only displays information about jobs belonging to the specified License Scheduler project.

-m *host_name*

Only displays jobs dispatched to the specified host.

-n *number_logfiles* | **-n 0**

Searches the specified number of event logs, starting with the current event log and working through the most recent consecutively numbered logs. The maximum number of logs you can search is 100. 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 will be inaccessible to `bhist`.

For example, if you specify 3, LSF searches `lsb.events`, `lsb.events.1`, and `lsb.events.2`. If you specify 4, LSF searches `lsb.events`, `lsb.events.1`, `lsb.events.2`, and `lsb.events.3`. However, if `lsb.events.2` is missing, both searches will include only `lsb.events` and `lsb.events.1`.

-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 `bhist` directly on an event log, you must specify a CPU factor.

Use `lsinfo` to get host model and CPU factor information.

-P *project_name*

Only displays information about jobs belonging to the specified project.

-q *queue_name*

Only displays information about jobs submitted to the specified queue.

-u *user_name* | **-u all**

Displays information about jobs submitted by the specified user, or by all users if the keyword `all` is specified.

job_ID | **"***job_ID***[***index***]"**

Searches all event log files and only displays information about the specified jobs. If you specify a job array, displays all elements chronologically.

This option overrides all other options except `-J`, `-N`, `-h`, and `-V`. When it is used with `-J`, only those jobs listed here that have the specified job name are displayed.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

Default format

Statistics of the amount of time that a job has 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 `sbatchd` on the execution host is temporarily unreachable).

TOTAL

The total time that the job has spent in all states; for a finished job, it is the turnaround time (that is, the time interval from job submission to job completion).

Long format (-l)

The `-l` option displays a long format listing with the following additional fields:

Project

The project the job was submitted from.

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.

FILES

Reads `lsb.events`.

SEE ALSO

`lsb.events`(5), `bgadd`(1), `bgdel`(1), `bjgroup`(1), `bsub`(1), `bjobs`(1), `lsinfo`(1)

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 is follows:

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*

Where:

- *year* is a four-digit number representing the calendar year.
- *month* is a number from 1 to 12, where 1 is January and 12 is December.
- *day* is a number from 1 to 31, representing the day of the month.
- *hour* is an integer from 0 to 23, representing the hour of the day on a 24-hour clock.
- *minute* is an integer from 0 to 59, representing the minute of the hour.
- . (period) represents the current month/day/hour:minute.
- .-*relative_int* is a number, from 1 to 31, specifying a relative start or end time prior to 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 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 a.m. and end at the last possible minute in February: February 28th at midnight.

ABSOLUTE TIME EXAMPLES

Assume the current time is May 9 17:06 2006:

`1,8` = May 1 00:00 2006 to May 8 23:59 2006

`,4` = the time of the first occurrence to May 4 23:59 2006

`6` = May 6 00:00 2006 to May 6 23:59 2006

`2/` = Feb 1 00:00 2006 to Feb 28 23:59 2006

`/12:` = May 9 12:00 2006 to May 9 12:59 2006

`2/1` = Feb 1 00:00 2006 to Feb 1 23:59 2006

`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 2006

`2001/12/31,2006/5/1` = from Dec 31, 2001 00:00:00 to May 1st 2006 23:59:59

RELATIVE TIME EXAMPLES

`.-9,` = April 30 17:06 2006 to the current time

`,.-2/` = the time of the first occurrence to Mar 7 17:06 2006

`.-9,.-2` = nine days ago to two days ago (April 30, 2006 17:06 to May 7, 2006 17:06)

bhosts

displays hosts and their static and dynamic resources

SYNOPSIS

bhosts [**-e** | **-l** | **-w**] [**-x**] [**-X**] [**-R** **"***res_req***"**] [*host_name* | *host_group*] ...

bhosts [**-e** | **-l** | **-w**] [**-X**] [**-R** **"***res_req***"**] [*cluster_name*]

bhosts [**-e**] **-s** [*shared_resource_name* ...]

bhosts [**-h** | **-V**]

DESCRIPTION

By default, returns the following information about all hosts: host name, host status, job state statistics, and job slot limits.

`bhosts` displays output for condensed host groups. These host groups are defined by `CONDENSE` in the `HostGroup` section of `lsb.hosts`. These host groups are displayed as a single entry with the name as defined by `GROUP_NAME` in the `HostGroup` section of `lsb.hosts`.

The `-l` and `-X` options display uncondensed output.

The `-s` option displays information about the numeric shared resources and their associated hosts.

With MultiCluster, displays the information about hosts available to the local cluster. Use -e to view information about exported hosts.

OPTIONS

-e

MultiCluster only. Displays information about resources that have been exported to another cluster.

-l

Displays host information in a (long) multi-line format. In addition to the default fields, displays information about the CPU factor, the current load, and the load thresholds.

Also displays information about the dispatch windows.

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

-w

Displays host information in wide format. Fields are displayed without truncation.

For condensed host groups, the `-w` option dispays 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. If a single host in the host group 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`, respectively.

For example, if there are five `ok`, two `unavail`, one `unreach`, and three `busy` hosts in a condensed host group `hg1`, its status is displayed as the following:

```
hg1 ok 5/2/1/3
```

If any hosts in the host group are closed, the status for the host group is displayed as `closed`, with no status for the other states:

```
hg1 closed
```

-x

Display hosts whose job exit rate has exceeded the threshold configured by EXIT_RATE in `lsb.hosts` for longer than JOB_EXIT_RATE_DURATION configured in `lsb.params`, and are still high. By default, these hosts will be closed the next time LSF checks host exceptions and invokes `eadmin`.

Use with the `-l` option to show detailed information about host exceptions.

If no hosts exceed the job exit rate, `bhosts -x` displays:

```
There is no exceptional host found
```

-X

Displays uncondensed output for host groups.

-R "*res_req*"

Only displays information about hosts that satisfy the resource requirement expression. For more information about resource requirements, see `lsfintro(1)`. The size of the resource requirement string is limited to 512 bytes.

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

-s [*shared_resource_name* ...]

Displays information about the specified shared resources. The resources must have numeric values. Returns the following information: the resource names, the total and reserved amounts, and the resource locations. If no shared resources are specified, displays information about all numeric shared resources.

host_name ... | *host_group* ...

Only displays information about the specified hosts. Do not use quotes when specifying multiple hosts.

For host groups, the names of the hosts belonging to the group are displayed instead of the name of the host group. Do not use quotes when specifying multiple host groups.

cluster_name

MultiCluster 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 batch jobs running and the host is removed from the configuration, the host name will be displayed as `lost_and_found`.

For condensed host groups, this is the name of host group.

STATUS

With MultiCluster, not shown for fully exported hosts.

The current status of the host and the `sbatchd` daemon. Batch jobs can only be dispatched to hosts with an `ok` status. The possible values for host status are as follows:

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 is not `ok`, `bhosts` displays the first host status it encounters as the overall status for the condensed host group. Use `bhosts -X` to see the status of indidual hosts in the host group.

unavail

The host is down, or LIM and `sbatchd` on the host are unreachable.

unreach

LIM on the host is running but `sbatchd` is unreachable.

closed

The host is not allowed to accept any remote batch jobs. There are several reasons for the host to be closed (see Host-Based `-l` Options).

unlicensed

The host does not have a valid LSF license.

JL/U

With MultiCluster, not shown for fully exported hosts.

The maximum number of job slots that the host can process on a per user basis. If a dash (-) is displayed, there is no limit.

For condensed host groups, this is the total number of job slots that all hosts in the host group can process on a per user basis.

The host will 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 that have slots reserved for them.

For preemptive scheduling, the accounting is different. These job slots are used by running jobs and by pending jobs that have slots reserved for them (see the description of PREEMPTIVE in `lsb.queues`(5) and JL/U in `lsb.hosts(5)`).

MAX

The maximum number of job slots available. If a dash (-) is displayed, there is no limit.

For condensed host groups, this is the total maximum number of job slots available in all hosts in the host group.

These job slots are used by running jobs, as well as by suspended or pending jobs that have slots reserved for them.

If preemptive scheduling is used, suspended jobs are not counted (see the description of PREEMPTIVE in `lsb.queues(5)` and MXJ in `lsb.hosts(5)`).

A host does not always have to allocate this many job slots if there are waiting jobs; the host must also satisfy its configured load conditions to accept more jobs.

NJOBS

The number of job slots used by jobs dispatched to the host. This includes running, suspended, and chunk jobs.

For condensed host groups, this is the total number of job slots used by jobs dispatched to any host in the host group.

RUN

The number of job slots used by jobs running on the host.

For condensed host groups, this is the total number of job slots used by jobs running on any host in the host group.

SSUSP

The number of job slots used by system suspended jobs on the host.

For condensed host groups, this is the total number of job slots used by system suspended jobs on any host in the host group.

USUSP

The number of job slots used by user suspended jobs on the host. Jobs can be suspended by the user or by the LSF administrator.

For condensed host groups, this is the total number of job slots used by user suspended jobs on any host in the host group.

RSV

The number of job slots used by pending jobs that have jobs slots reserved on the host.

For condensed host groups, this is the total number of job slots used by pending jobs that have job slots reserved on any host in the host group.

Host-Based -l Option

In addition to the above fields, the `-l` option also displays the following:

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.

If the host's operating system supports checkpoint copy, this is indicated here. With checkpoint copy, the operating system automatically copies all open files to the checkpoint directory when a process is checkpointed. Checkpoint copy is currently supported only on Cray systems.

STATUS

The long format shown by the `-l` option gives the possible reasons for a host to be closed:

closed_Adm

The host is closed by the LSF administrator or `root` (see `badmin(8)`). No job can be dispatched to the host, but jobs that are executing on the host will not be affected.

closed_Lock

The host is locked by the LSF administrator or `root` (see `lsadmin(8)`). All batch jobs on the host are suspended by LSF.

closed_Wind

The host is closed by its dispatch windows, which are defined in the configuration file `lsb.hosts(5)`. Jobs already started are not affected by the dispatch windows.

closed_Full

The configured maximum number of batch job slots on the host has been reached (see MAX field below).

closed_Excl

The host is currently running an exclusive job.

closed_Busy

The host is overloaded, because some load indices go beyond the configured thresholds (see `lsb.hosts(5)`). The displayed thresholds that cause the host to be busy are preceded by an asterisk (*).

closed_LIM

LIM on the host is unreachable, but `sbatchd` is ok.

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 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 `bqueues(1)`.

CURRENT LOAD

Displays the total and reserved host load.

Reserved

You specify reserved resources by using `bsub -R` (see `lsfintro(1)`). These resources are reserved by jobs 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 utilization, 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 seen by `lsload(1)`.

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 seen by `lsload(1)`.

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 (see `bqueues(1)`) and `lsb.queues(5)`). For an explanation of the thresholds and load indices, see the description for the "QUEUE SCHEDULING PARAMETERS" keyword under the `-l` option in `bqueues(1)`.

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 LSF administrator specified an administrator comment with the `-C` option of the `badmin` host control commands `hclose` or `hopen`, the comment text is displayed.

Resource-Based -s Option

The `-s` option displays the following: the amounts used for scheduling, the amounts reserved, and the associated hosts for the shared resources. Only shared resources with numeric values are displayed. See `lim(8)`, and `lsf.cluster(5)` on how to configure shared resources.

The following fields are displayed:

RESOURCE

The name of the resource.

TOTAL

The value of the shared resource used for scheduling. This is the sum of the current and the reserved load for the shared resource.

RESERVED

The amount reserved by jobs. You specify the reserved resource using `bsub -R` (see `lsfintro(1)`).

LOCATION

The hosts that are associated with the shared resource.

FILES

Reads `lsb.hosts`.

SEE ALSO

`lsb.hosts`(5), `bqueues`(1), `lsfintro`(1), `lshosts`(1), `badmin`(8), `lsadmin`(8)

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 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 (`/`) (see `bmgroup`(1)).

USER/GROUP

Name of users or user groups who have access to the host partition (see `bugroup`(1)).

SHARES

Number of shares of resources assigned to each user or user group in this host partition, as configured in the file `lsb.hosts`. 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, and a lower CPU_TIME and RUN_TIME will have higher PRIORITY.

STARTED

Number of job slots used by running or suspended jobs owned by users or user groups in the host partition.

RESERVED

Number of job slots reserved by the jobs owned by users or user groups in the host partition.

CPU_TIME

Cumulative CPU time used by jobs of users or user groups executed in the host partition. Measured in seconds, to one decimal place.

LSF calculates the cumulative CPU time using the actual (not normalized) CPU time and a decay factor such that 1 hour of recently-used CPU time decays to 0.1 hours after an interval of time specified by HIST_HOURS in `lsb.params` (5 hours by default).

RUN_TIME

Wall-clock run time plus historical run time of jobs of users or user groups that are executed in the host partition. Measured in seconds.

LSF calculates the historical run time using the actual run time of finished jobs and a decay factor such that 1 hour of recently-used run time decays to 0.1 hours after an interval of time specified by HIST_HOURS in `lsb.params` (5 hours by default). Wall-clock run time is the run time of running jobs.

FILES

Reads `lsb.hosts`.

SEE ALSO

`bugroup`(1), `bmgroup`(1), `lsb.hosts`(5)

bjgroup

displays information about job groups

SYNOPSIS

bjgroup [**-s**]

bjgroup [**-h** | **-V**]

DESCRIPTION

Displays all job groups.

OPTIONS

-s

Sorts job groups by hierarchy. For example, for job groups named `/A`, `/A/B`, `/X` and `/X/Y`, `bjgroup` without `-s` displays:

```
% bjgroup
GROUP_NAME          NJOBS    PEND     RUN      SSUSP   USUSP   FINISH
/A                  0        0        0        0       0       0
/X                  0        0        0        0       0       0
/A/B                0        0        0        0       0       0
/X/Y                0        0        0        0       0       0
```

For the same job groups, `bjgroup -s` displays:

```
% bjgroup -s
GROUP_NAME          NJOBS    PEND     RUN      SSUSP   USUSP   FINISH
/A                  0        0        0        0       0       0
/A/B                0        0        0        0       0       0
/X                  0        0        0        0       0       0
/X/Y                0        0        0        0       0       0
```

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

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 job slots used by jobs in the specified service class. A parallel job is counted as 1 job, regardless of the number of job slots it will use.

PEND

The number of pending job slots used by jobs in the specified job group.

RUN

The number of job slots used by running jobs in the specified job group.

SSUSP

The number of job slots used by the system-suspended jobs in the specified job group.

USUSP

The number of job slots used by user-suspended jobs in the specified job group.

FINISH

The number of jobs in the specified job group in EXITED or DONE state.

EXAMPLE

```
% bjgroup
GROUP_NAME           NJOBS    PEND     RUN     SSUSP  USUSP  FINISH
/fund1_grp            5        4        0       1      0       0
/fund2_grp            11       2        5       0      0       4
/bond_grp             2        2        0       0      0       0
/risk_grp             2        1        1       0      0       0
/admi_grp             4        4        0       0      0       0
```

SEE ALSO

`bgadd`(1), `bgdel`(1)

bjobs

displays information about LSF jobs

SYNOPSIS

bjobs [**-a**] [**-A**] [**-w** | **-l**] [**-X**] [**-g** *job_group_name* |**-sla** *service_class_name*] [**-J** *job_name*] [**-Lp** *ls_project_name*] [**-m** *host_name* | **-m** *host_group* | **-m** *cluster_name*] [**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*] [**-P** *project_name*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] [**-x**] *job_ID* | **"***job_ID***[***index_list***]"** ...

bjobs [**-d**] [**-p**] [**-r**] [**-s**] [**-A**] [**-w** | **-l**] [**-X**] [**-g** *job_group_name* |**-sla** *service_class_name*] [**-J** *job_name*] [**-Lp** *ls_project_name*] [**-m** *host_name* | **-m** *host_group* | **-m** *cluster_name*] [**-N** *host_name* | **-N** *host_model* | **-N** *CPU_factor*] [**-P** *project_name*] [**-q** *queue_name*] [**-u** *user_name* | **-u** *user_group* | **-u all**] [**-x**] *job_ID* | **"***job_ID***[***index_list***]"** ...

bjobs [**-h** | **-V**]

DESCRIPTION

By default, displays information about your own pending, running and suspended jobs.

`bjobs` displays output for condensed host groups. These host groups are defined by `CONDENSE` in the `HostGroup` section of `lsb.hosts`. These host groups are displayed as a single entry with the name as defined by `GROUP_NAME` in the `HostGroup` section of `lsb.hosts`.

If you defined `LSB_SHORT_HOSTLIST=1` in `lsf.conf`, parallel jobs running in the same condensed host group are displayed as an abbreviated list.

The `-l` and `-X` options display uncondensed output.

To display older historical information, use `bhist`.

Exit code 0 is returned for all job states.

OPTIONS

-a

Displays information about jobs in all states, including finished jobs that finished recently, 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).

-A

Displays summarized information about job arrays. 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.

-d

Displays information about jobs that finished recently, within an interval specified by CLEAN_PERIOD in `lsb.params` (the default period is 1 hour).

-l

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

The `-l` option displays the following additional information: project name, job command, current working directory on the submission host, pending and suspending reasons, job status, resource usage, resource usage limits information, runtime resource usage information on the execution hosts.

Use `bjobs -A -l` to display detailed information for job arrays including job array job limit (`%`*`job_limit`*) if set.

If JOB_IDLE 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 created with the `brsvadd` command, `bjobs -l` shows the reservation ID used by the job.

If LSF_HPC_EXTENSIONS="SHORT_PIDLIST" is specified in `lsf.conf`, 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. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

-p

Displays pending jobs, together with the pending reasons that caused each job not to be dispatched during the last dispatch turn. The pending reason shows the number of hosts for that reason, or names the hosts if `-l` is also specified.

With MultiCluster, -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.

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.
```

-r

Displays running jobs.

-s

Displays suspended jobs, together with the suspending reason that caused each job to become suspended.

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 will be updated according to the load index. The reasons could be as old as the time interval specified by SBD_SLEEP_TIME in `lsb.params`. So the reasons shown may not reflect the current load situation.

-w

Wide format. Displays job information without truncating fields.

-X

Displays uncondensed output for host groups.

-g *job_group_name*

Displays information about jobs attached to the job group specified by *job_group_name*. For example:

```
% 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
```

You cannot use `-g` with `-sla`. A job can either be attached to a job group or a service class, but not both.

`bjobs -l` with `-g` displays the full path to the group to which a job is attached. For example:

```
% 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: Submitted from host <hostA>, CWD
</home/user1;
Tue Jun 17 16:22:01: Started on <hostA>;
...
```

-J *job_name*

Displays information about the specified jobs or job arrays.

-Lp *ls_project_name*

Displays jobs that belong to the specified LSF License Scheduler project.

-m *host_name* ... | **-m** *host_group* ... | **-m** *cluster_name* ...

Only displays jobs dispatched to the specified hosts. To see the available hosts, use `bhosts`.

If a host group is specified, displays jobs dispatched to all hosts in the group. To determine the available host groups, use `bmgroup`.

With MultiCluster, displays jobs in the specified cluster. If a remote cluster name is specified, you will see the remote job ID, even if the execution host belongs to the local cluster. To determine the available clusters, use `bclusters`.

-N *host_name* | **-N** *host_model* | **-N** *CPU_factor*

Displays the normalized CPU time consumed by the job. Normalizes using the CPU factor specified, or the CPU factor of the host or host model specified.

-P *project_name*

Only displays jobs that belong to the specified project.

-q *queue_name*

Only displays jobs in the specified queue.

The command `bqueues` returns a list of queues configured in the system, and information about the configurations of these queues.

In MultiCluster, you cannot specify remote queues.

-sla *service_class_name*

Displays jobs belonging to the specified service class.

Use `bsla` to display the properties of service classes configured in `LSB_CONFDIR/`*`cluster_name`*`/configdir/lsb.serviceclasses` (see `lsb.serviceclasses`(5)) and dynamic information about the state of each configured service class.

You cannot use `-g` with `-sla`. A job can either be attached to a job group or a service class, but not both.

-u *user_name...* | **-u** *user_group...* | **-u all**

Only displays jobs that have been submitted by the specified users. The keyword `all` specifies all users.

-x

Displays unfinished jobs that have triggered a job exception (overrun, underrun, idle). 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 | **"***job_ID***[***index***]"**

Displays information about the specified jobs or job arrays.

If you use `-A`, specify job array IDs without the index list.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

Pending jobs are displayed in the order in which they will be 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 will be 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.

Default Display

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 will be 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 MultiCluster 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 MultiCluster, 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, the host name is displayed as the name of the condensed host 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 (see `bsub` (1)). If the job name is too long to fit in this field, then only the latter part of the job name is displayed.

SUBMIT_TIME

The submission time of the job.

-l output

The `-l` option displays a long format listing with the following additional fields:

Project

The project the job was submitted from.

Command

The job command.

CWD

The current working directory on the submission host.

PENDING REASONS

The reason the job is in the PEND or PSUSP state. The names of the hosts associated with each reason will be 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.

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 job has been suspended by LSF due to either of the following two causes:

- 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 will become 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.
- The host on which a rerunnable job is running is unavailable and the job has been requeued by LSF with a new job ID, as if the job were submitted as a new job.

After the execution host becomes available, LSF will try to kill the ZOMBI job. Upon successful termination of the ZOMBI job, the job's status will be changed to EXIT.

With MultiCluster, when a job running on a remote execution cluster becomes a ZOMBI job, the execution cluster will treat 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.

RESOURCE USAGE

For the MultiCluster job forwarding model, this information is not shown if MultiCluster resource usage updating is disabled.

The values for the current usage of a job include:

CPU time

Cumulative total CPU time in seconds of all processes in a job.

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, in MB.

SWAP

Total virtual memory usage of all processes in a job, in MB.

NTHREAD

Number of currently active threads of a job.

PGID

Currently active process group ID in a job.

PIDs

Currently active processes in a job.

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
- PROCLIMIT
- MEMLIMIT
- SWAPLIMIT
- PROCESSLIMIT
- THREADLIMIT

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.

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.

Job Array Summary Information

If you use `-A`, displays 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.

EXAMPLES

```
% bjobs -pl
```

Displays detailed information about all pending jobs of the invoker.

```
% bjobs -ps
```

Display only pending and suspended jobs.

```
% 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`.

```
% 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 host groups.

```
% bjobs -sla Uclulet
```

Displays all jobs belonging to the service class `Uclulet`.

SEE ALSO

bsub(1), bkill(1), bhosts(1), bmgroup(1), bclusters(1), bqueues(1), bhist(1), bresume(1), bsla(1), bstop(1), lsb.params(5), lsb.serviceclasses(5), mbatchd(8)

bkill

sends signals to kill, suspend, or resume unfinished jobs

SYNOPSIS

bkill [**-l**] [**-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**] [**-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**] [*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 are sent to give the job a chance to clean up before termination, then SIGKILL is sent to kill the job. The time interval between sending each signal is defined by the JOB_TERMINATE_INTERVAL parameter in `lsb.params(5)`.

You must specify a job ID or `-g`, `-J`, `-m`, `-u`, or `-q`. Specify job ID `0` (zero) to kill multiple jobs.

On Windows, job control messages replace the SIGINT and SIGTERM signals (but only customized applications can process them) and the `TerminateProcess()` system call is sent to kill the job.

Exit code 130 is returned when a dispatched job is killed with `bkill`.

Users can only operate on their own jobs. Only `root` and LSF administrators can operate on jobs 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, `bkill` 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)

Using `bkill` on a repetitive job kills the current run, if the job has been started, and requeues the job. See `bcadd`(1) and `bsub`(1) for information on setting up a job to run repetitively.

If the job cannot be killed, use `bkill -r` 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 (`-g`, `-m`, `-q`, `-u`, and `-J`).

-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 CLEAN_PERIOD in `lsb.params`. Jobs killed in this manner are not logged to `lsb.acct`.

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, `bkill` 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 supported by `bkill`. This is a subset of signals supported by `/bin/kill` and is platform-dependent.

-r

Removes a job from the LSF system without waiting for the job to terminate in the operating system.

Sends the same series of signals as `bkill` without `-r`, except that the job is removed from the system immediately, the job is marked as EXIT, and the job resources that LSF monitors are released as soon as LSF receives the first signal.

Also operates on jobs for which a `bkill` command has been issued but which cannot be reached to be acted on by `sbatchd` (jobs in ZOMBI state). If `sbatchd` recovers before the jobs are completely removed, LSF ignores the zombi jobs killed with `bkill -r`.

Use `bkill -r` only on jobs that cannot be killed in the operating system, or on jobs that cannot be otherwise removed using `bkill`.

The `-r` option cannot be used with the `-s` option.

-g *job_group_name*

Operates only on jobs in the job group specified by *job_group_name*.

You cannot use `-g` with `-sla`. A job can either be attached to a job group or a service class, but not both.

`bkill` 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 only kills jobs in `/risk_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.

-m *host_name* | **-m** *host_group*

Operates only on jobs 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. See `bhosts(1)` and `bmgroup(1)` for more 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.

See `bqueues`(1) for more 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 signal names are listed by `bkill -l`.

The `-s` option cannot be used with the `-r` option.

Use `bkill -s` to suspend and resume jobs by using the appropriate signal instead of using `bstop` or `bresume`. Sending the SIGCONT signal is the same as using `bresume`.

Sending the SIGSTOP signal to sequential jobs or the SIGTSTP to parallel jobs is the same as using `bstop`.

You cannot suspend a job that is already suspended, or resume a job that is not suspended. Using SIGSTOP or SIGTSTP on a job that is in the USUSP state has no effect and using SIGCONT on a job that is not in either the PSUSP or the USUSP state has no effect. See `bjobs(1)` for more information about job states.

-sla *service_class_name*

Operates on jobs belonging to the specified service class.

If *job_ID* is not specified, only the most recently submitted job is operated on.

You cannot use -g with -sla. A job can either be attached to a job group or a service class, but not both.

The -sla option is ignored if a job ID other than 0 is specified in the *job_ID* option.

Use bsla to display the properties of service classes configured in LSB_CONFDIR/*cluster_name*/configdir/lsb.serviceclasses (see lsb.serviceclasses(5)) and dynamic information about the state of each configured service class.

-u *user_name* | **-u** *user_group* | **-u all**

Operates only on jobs submitted by the specified user or user group, or by all users if the reserved user name all is specified.

If *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 (see bjobs(1)). For job arrays, quotation marks must enclose the job ID and index, and index must be enclosed in square brackets.

Jobs 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) 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 (see bsub(1)) and may be obtained with the bjobs command (see bjobs(1)).

-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
```

Kill all jobs belonging to the service class named Tofino.

```
% bkill -g /risk_group 0
```

Kills all jobs in the job group /risk_group.

SEE ALSO

bsub(1), bjobs(1), bqueues(1), bhosts(1), bresume(1), bsla(1), bstop(1), bgadd(1), bgdel(1), bjgroup(1), bparams(5), lsb.serviceclasses(5), mbatchd(8), kill(1), signal(2)

bladmin

reconfigures the Platform LSF License Scheduler daemon (`bld`).

SYNOPSIS

bladmin reconfig | **shutdown**

bladmin [**-h** | **-V**]

DESCRIPTION

Use this command to reconfigure the License Scheduler daemon (`bld`).

You must be a License Scheduler administrator to use this command.

OPTIONS

reconfig [*host_name* ... | **all**]

Reconfigures License Scheduler.

shutdown [*host_name* ... | **all**]

Shuts down License Scheduler.

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

SEE ALSO

`blhosts`(1), `lsf.licensescheduler`(5), `lsf.conf`(5)

blcollect

license information collection daemon

SYNOPSIS

blcollect [**-c** *collector_name*] [**-m** *host_name* ...] [**-p** *license_scheduler_port*]

blcollect [**-h** | **-V** | **-i** *lmstat_interval* | **-D** *lmstat_path*]

DESCRIPTION

Periodically collects license usage information from Macrovision® FLEXnet™. It queries FLEXnet for license usage information from the FLEXnet `lmstat` command, and passes the information to the 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 FLEXnet on one host. Use `blcollect` to distribute the license collection on multiple hosts.

For each service domain configuration in `lsf.licensescheduler`, specify one name for `blcollect` 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 `blcollect`.

OPTIONS

-c

Mandatory. Specify the collector name you set in `lsf.licensescheduler`. You must use the collector name (`LIC_COLLECT`) you define in the `ServiceDomain` section of the configuration file.

-m

Mandatory. Specifies a space-separated list of hosts to which license information is sent. The hosts do not need to be running License Scheduler or a FLEXnet. Use fully qualified host names.

-p

Mandatory. Corresponds to the License Scheduler listening port, which is set in `lsf.licensescheduler`. The default value is 9581.

-i *lmstat_interval*

Optional. The frequency in seconds of the calls that License Scheduler makes to `lmstat` to collect license usage information from FLEXnet.

The default interval is 60 seconds.

-D *lmstat_path*

Optional. Location of the FLEXnet command `lmstat`.

-h

Prints command usage to `stderr` and exits.

-v

Prints release version to `stderr` and exits.

SEE ALSO

`lsf.licensescheduler`(5)

blhosts

displays the names of all the hosts running the License Scheduler daemon (bld).

SYNOPSIS

blhosts [**-h** | **-V**]

DESCRIPTION

Displays a list of hosts running the License Scheduler daemon. This includes the License Scheduler master host and all the candidate License Scheduler hosts running bld.

OPTIONS

-h

Prints command usage to stderr and exits.

-V

Prints release version to stderr and exits.

OUTPUT

Prints out the names of all the hosts running the License Scheduler daemon (bld).

For example, the following sample output shows the License Scheduler master host and two candidate License Scheduler hosts running bld:

```
bld is running on:
master: host1.domain1.com
slave: host2.domain1 host3.domain1
```

SEE ALSO

blinfo(1) blstat(1) bladmin(8)

blimits

displays information about resource allocation limits of running jobs

SYNOPSIS

blimits [**-n** *limit_name* ...] [**-m** *host_name* | **-m** *host_group* | **-m** *cluster_name* ...] [**-P** *project_name* ...] [**-q** *queue_name* ...] [**-u** *user_name* | **-u** *user_group* ...]

blimits -c

blimits -h | **-V**

DESCRIPTION

Displays current usage of resource allocation limits configured in Limit sections in `lsb.resources`:

- Configured limit policy name
- Users (`-u` option)
- Queues (`-q` option)
- Hosts (`-m` option)
- Project names (`-P` option)

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 `blimits` displays the limit for SLOTS as 3/10.

Note that if there are no jobs running against resource allocation limits, LSF indicates that there is no information to be displayed:

```
No resource usage found.
```

If limits MEM, SWP, or TMP are configured as percentages, both the limit and the amount used are displayed in MB. For example, `lshosts` displays maxmem of 249 MB, and MEM is limited to 10% of available memory. If 10 MB out of are used, `blimits` displays the limit for MEM as 10/25 (10 MB USED from a 25 MB LIMIT).

Configured limits and resource usage for builtin resources (slots, mem, tmp, and swp load indices) are displayed as INTERNAL RESOURCE LIMITS separately from custom external resources, which are shown as EXTERNAL RESOURCE LIMITS.

Limits are displayed for both the vertical tabular format and the horizontal format for Limit sections. Since a vertical format Limit section has no name, `blimits` displays NONAME*nnn* under the NAME column for these limits, where the unnamed limits are numbered in the order 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` commands displays these as MXJ limits.

In MultiCluster, `blimits` returns the information about all limits in the local cluster.

Limit names and policies are set up by the LSF administrator. See `lsb.resources`(5) for more information.

OPTIONS

-c

Displays all resource configurations in `lsb.resources`. This is the same as `bresources` with no options.

-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 (’).

-m *host_name* | **-m** *host_group* | **-m** *cluster_name ...*

Displays resource allocation limits for the specified hosts. Do not use quotes when specifying multiple hosts.

To see the available hosts, use `bhosts`.

For host groups:

- If the limits are configured with HOSTS, the name of the host group is displayed.
- If the limits are configured with PER_HOST, the names of the hosts belonging to the group are displayed instead of the name of the host group.

PER_HOST slot limits are not displayed. The bhosts command displays these as MXJ limits.

For a list of host groups see `bmgroup(1)`.

In MultiCluster, if a cluster name is specified, displays resource allocation limits in the specified cluster.

-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 configured in the system, and information about the configurations of these queues.

In MultiCluster, 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. For a list of user groups see `bugroup(1)`).

-h

Prints command usage to `stderr` and exits.

-v

Prints LSF release version to `stderr` and exits.

OUTPUT

Configured limits and resource usage for builtin resources (slots, mem, tmp, and swp load indices) are displayed as INTERNAL RESOURCE LIMITS separately from custom external resources, which are shown as EXTERNAL RESOURCE LIMITS.

Resource Consumers

`blimits` 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. See `bugroup(1)`.

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 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 MultiCluster 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`.

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. See `bmgroup(1)`.

PER_HOST slot limits are not displayed. The bhosts command displays these as MXJ limits.

PROJECTS

List of project names on which limits are enforced., as specified by the Limit section parameters PROJECTS or PER_PROJECT.

Resource Limits

`blimits` displays resource allocation limits for the following resources:

SLOTS

Number of slots currently used and maximum number of slots configured for the limit policy, as specified by the Limit section SLOTS parameter.

MEM

Amount of memory currently used and maximum configured for the limit policy, as specified by the Limit section MEM parameter.

TMP

Amount of tmp space currently used and maximum amount of tmp space configured for the limit policy, as specified by the Limit section TMP parameter.

SWP

Amount of swap space currently used and maximum amount of swap space configured for the limit policy, as specified by the Limit section SWP parameter.

EXAMPLE

The following command displays limit configuration and dynamic usage information for project `proj1`:

```
%blimits -P proj1
INTERNAL RESOURCE LIMITS:

NAME       USERS       QUEUES      HOSTS      PROJECTS    SLOTS    MEM     TMP    SWP
limit1     user1            -      hostA         proj1      2/6     -       -       -
limit2       -              -      hostB   proj1 proj2      1/3     -       -        -

EXTERNAL RESOURCE LIMITS:

NAME       USERS       QUEUES      HOSTS      PROJECTS      tmp1
limit1     user1            -      hostA proj1       1/1
```

SEE ALSO

`bclusters`(1), `bhosts`(1), `bhist`(1), `bmgroup`(1), `bqueues`(1), `bugroup`(1), `lsb.resources`(5)

blinfo

displays static License Scheduler configuration information.

SYNOPSIS

blinfo [**-a** | **-Lp** | **-p** | **-D** | **-G**]

blinfo [**-h** | **-V**]

DESCRIPTION

Displays different license configuration information, depending on the option selected.

By default, displays information about the distribution of licenses managed by License Scheduler.

OPTIONS

-a

Prints out all information, including information about non-shared licenses (NON_SHARED_DISTRIBUTION) and workload distribution (WORKLOAD_DISTRIBUTION).

`blinfo -a` does not display NON_SHARED inforation for hierarchical project group scheduling policies. Use `blinfo -G` to see hierarchical group configuration.

-D

Lists the License Scheduler service domains and the corresponding FLEXnet license server hosts.

-G

Lists the hierarchical configuration information.

-Lp

Lists the projects managed by License Scheduler.

If PRIORITY is defined in the `Projects` Section of `lsf.licensescheduler`, this option also lists the priorities of each project.

-p

Displays values of `lsf.licensescheduler` configuration parameters. This is useful for troubleshooting.

-h

Prints command usage to `stderr` and exits.

-V

Prints the License Scheduler release version to `stderr` and exits.

OUTPUT

Default Output

Displays the following fields:

FEATURE

The license name. This becomes the license token name.

SERVICE_DOMAIN

The name of the service domain that provided the license.

TOTAL

The total number of licenses 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*]]. This determines how many licenses a project is entitled to use when there is competition for licenses. The percentage is calculated from the share specified in the configuration file.

Project Output (-Lp)

List of License Scheduler projects.

PROJECT

The project name.

PRIORITY

The priority of the project if it is different from the default behaviour. A larger number indicates a higher priority.

Service Domain Output (-D)

SERVICE_DOMAIN

The service domain name.

LIC_SERVERS

Names of FLEXnet license server hosts that belong the to service domain. Each host name is enclosed in parentheses, as shown:

(*port_number***@***host_name***)**

Redundant hosts (that share the same FLEXnet license file) are grouped together as shown:

(*port_number***@***host_name port_number***@***host_name port_number***@***host_name***)**

Parameters Output (-p)

ADMIN

The License Scheduler administrator

HOSTS

License Scheduler candidate hosts.

LICENSE_FILE

Location of the License Scheduler license file.

PORT

TCP listening port used by License Scheduler.

Hierarchical Output (-G)

The following fields describe the values of their corresponding configuration fields in the `ProjectGroup` Section of `lsf.licensescheduler`.

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 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.

All Output (-a)

Same as Default Output with NON_SHARED_DISTRIBUTION.

NON-SHARED_DISTRIBUTION

This column is displayed directly under `DISTRIBUTION` with the `-a` option. If there are non-shared licenses, then the non-shared license information is output in the following format: [*project_name*, *number_licenses_non_shared*]

If there are no non-shared licenses, then the following license information is output - (dash)

EXAMPLES

- `blinfo -a` displays both NON_SHARED_DISTRIBUTION and WORKLOAD_DISTRIBUTION information:

```
% 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%]
```

- `blinfo -a` does not display NON_SHARED_DISTRIBUTION, if the NON_SHARED_DISTRIBUTION is not defined:

```
% blinfo -a
FEATURE       SERVICE_DOMAIN  TOTAL  DISTRIBUTION
g1            LS              0      [p1, 50.0%] [p2, 50.0%]
                                     WORKLOAD_DISTRIBUTION
                                     [LSF 66.7%, NON_LSF 33.3%]
g2            LS              0      [p1, 50.0%] [p2, 50.0%]
g33           WS              0      [p1, 50.0%] [p2, 50.0%]
```

- `blinfo -a` does not display WORKLOAD_DISTRIBUTION, if the WORKLOAD_DISTRIBUTION is not defined:

```
% blinfo -a
FEATURE       SERVICE_DOMAIN  TOTAL  DISTRIBUTION
g1            LS              3      [p1, 50.0%] [p2, 50.0% / 2]
                                     NON_SHARED_DISTRIBUTION
                                     [p2, 2]
```

FILES

Reads `lsf.licensescheduler`

SEE ALSO

`blstat`(1), `blusers`(1)

blkill

terminates an interactive License Scheduler task

SYNOPSIS

blkill [**-t** *seconds*] *task_ID*

blkill [**-h** | **-V**]

DESCRIPTION

Terminates a running or waiting interactive task in License Scheduler.

Users can kill their own tasks. You must be a License Scheduler administrator to terminate another user's task.

By default, `blkill` notifies the user and waits 30 seconds before killing the task.

OPTIONS

task_ID

Task ID of the task you want to kill.

-t *seconds*

Specify how many seconds to delay before killing 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 License Scheduler release version to `stderr` and exits.

blstat

displays dynamic license information

SYNOPSIS

blstat [**-G**] [**-s**] [**-S**] [**-D** *service_domain_name* | **"***service_domain_name* ...**"**]
[**-Lp** *ls_project_name* | **"***ls_project_name* ...**"**]
[**-t** *token_name* | **"***token_name* ...**"**]

blstat [**-h** | **-V**]

DESCRIPTION

Displays license usage statistics.

By default, shows information about all licenses and all clusters.

OPTIONS

-D *service_domain_name* | **"***service_domain_name* ...**"**

Only shows information about specified service domains. Use spaces to separate multiple names, and enclose them in quotation marks.

-G

Displays dynamic hierarchical license information.

`blstat -G` also works with the `-t` option to only display hierarchical information for the specified feature names.

-Lp *ls_project_name* | **"***ls_project_name* ...**"**

Only shows license information for specified projects. Use spaces to separate multiple names, and enclose them in quotation marks.

-s

Displays license usage of the LSF and non-LSF workloads. Workload distributions are defined by WORKLOAD_DISTRIBUTION in `lsf.licensescheduler`. If there are any distribution policy violations, `blstat` marks these with an asterisk (`*`) at the beginning of the line.

-S

Displays information on the license server associated with license features.

-t *token_name* | **"***token_name* ...**"**

Only shows information about specified licenses. Use spaces to separate multiple names, and enclose them in quotation marks.

-h

Prints command usage to `stderr` and exits.

-V

Prints the release version to `stderr` and exits.

OUTPUT

Information is organized first by license feature, then by service domain. For each combination of license and service domain, License Scheduler displays a line of summary information followed by rows of license project information (one row for each license project configured to use the license).

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.

Summary output

FEATURE

The license name. (This 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 License Scheduler projects. (Licenses in use have been checked out from the FLEXnet license manager.)

TOTAL_RESERVE

The number of licenses reserved for License Scheduler projects. (Licenses that are reserved and have not been checked out from the FLEXnet license manager.)

TOTAL_FREE

The number of free licenses that are available to License Scheduler projects. (Licenses that are not reserved or in use.)

OTHERS

The number of licenses checked out by users who are not submitting their jobs to License Scheduler projects.

By default, these licenses are not being managed by License Scheduler policies.

To enforce license distribution policies for these license features, configure ENABLE_DYNAMIC_RUSAGE=Y in the feature section for those features in `lsf.licensescheduler`.

Workload output

LSF_USE

The total number of licenses in use by License Scheduler projects in the LSF workload.

LSF_DESERVE

The total number of licenses assigned to License Scheduler projects in the LSF workload..

LSF_FREE

The total number of free licenses available to 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 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 each project that is configured to use the license, `blstat` displays the following information.

PROJECT

The License Scheduler project name.

SHARE

The percentage of licenses assigned to the license project by the License Scheduler administrator. This determines how many licenses the project is entitled to when there is competition for licenses. This information is static.

The percentage is calculated to one decimal place using the share assignment in `lsf.licensescheduler`.

INUSE

The number of licenses in use by the license project. (Licenses in use have been checked out from the FLEXnet license manager.)

RESERVE

The number of licenses reserved for the license project. (The corresponding job has started to run, but has not yet checked out its license from the FLEXnet license manager.)

FREE

The number of licenses the license project has free. (The license tokens have been allocated to the license project by License Scheduler, but the licenses are not reserved and have not yet been checked out from the FLEXnet license manager.)

DEMAND

`y|n`

`n` indicates that this license project's license requirements are satisfied.

`y` indicates that this license project needs more license tokens. (It has pending jobs waiting for this license feature.)

NON_SHARED

The number of non-shared licenses belonging to the license project. (The license tokens allocated to non-shared distribution are scheduled before the tokens allocated to shared distribution.)

Hierarchical output

SHARE_INFO_FOR

The root member and name of the hierarchical group. The project information displayed after this title shows the information specific to this particular hierarchical group. If this root member is itself a member of another hierarchical group, the relationship is displayed as follows:

/*root_name***/***member_name***/**...

PROJECT/GROUP

The members of the hierarchical group, listed by its group or project name.

SEE ALSO

`blhosts`(1), `blinfo`(1)

bltasks

displays 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 managed by License Scheduler (submitted using `taskman`).

By default, displays information about all tasks.

OPTIONS

task_ID

Only displays information about the specified task.

-l

Long format. Displays detailed information for each task in a multi-line format.

-p

Only displays information about tasks with `PREEMPTED` status.

Cannot be used with `-r` or `-w`.

-r

Only displays information about tasks with `RUN` status.

Cannot be used with `-p` or `-w`.

-w

Only displays information about tasks with `WAIT` status.

Cannot be used with `-p` or `-r`.

-Lp "*ls_project_name...*"

Only displays information about tasks associated with the specified projects.

-m "*host_name...*"

Only displays information about tasks submitted from the specified hosts.

-t "*terminal_name...*"

Only displays information about tasks submitted from the specified terminals.

-u "*user_name...*"

Only displays information about tasks submitted by the specified users.

-h

Prints command usage to `stderr` and exits.

-v

Prints License Scheduler release version to `stderr` and exits.

OUTPUT

Default Output

Displays the short format with the following information:

TID

Task ID that License Scheduler assigned to the task.

USER

The user who submitted the task.

STAT

The current status of the task.

- RUN: Task is running.

- WAIT: Task has not yet started.

- PREEMPT: Task has been 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 License Scheduler token.

CONNECT TIME

The submission time of the task.

Output for –l Option

Displays detailed information for each task in multi-line format. If the task is in WAIT status, `bltasks` displays "The application manager is waiting for a token to start" and the resource requirement. Otherwise, the current resource usage of task is displayed as follows:

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 License Scheduler task manager is executing.

blusers

displays license usage information

SYNOPSIS

blusers [**-J** | **-l** | **-P -j** *job_ID* **-u** *user_name* **-m** *host_name* | **-P -c** *cluster_name* **-j** *job_ID* **-u** *user_name* **-m** *host_name* | **-h** | **-V**]

DESCRIPTION

By default, displays summarized information about usage of licenses.

OPTIONS

-J

Displays detailed license usage information about each job.

-l

Long format. Displays additional license usage information. See "OUTPUT" for a description of information that is displayed.

-P -j *job_ID* **-u** *user_name* **-m** *host_name*
-P -c *cluster_name* **-j** *job_ID* **-u** *user_name* **-m** *host_name*

This string of options is designed to be used in a customized preemption script. To identify a job, specify the LSF job ID, the user name, and the name of the host where the job is running.

(If the job is an interactive task submitted using `taskman`, do not specify -c *cluster_name*.)

You will see the display terminal used by the job, the licenses it has checked out, and the license servers that provided the licenses. There is one line of output for each license feature from each FLEXnet license server, in the format:

port_number@host_name token_name user_name host_name display

-h

Prints command usage to `stderr` and exits.

-V

Prints License Scheduler release version to `stderr` and exits.

OUTPUT

Default Output

FEATURE

The license name. This 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 have started.

NLICS

The number of licenses checked out from FLEXnet.

NTASKS

The number of running tasks using these licenses.

-J Output

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 has been 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 “-” for an interactive job.

START_TIME

The job start time.

Displays the following information for each license in use by the job:

RESOURCE

The name of the license requested by the job.

RUSAGE

The number of licenses requested by the job.

SERVICE_DOMAIN

The name of the service domain that provided the license.

The keyword UNKNOWN means the job requested a license from License Scheduler but has not checked out the license from FLEXnet.

Long Output (-l)

Displays the default output and the following additional information for each job:

OTHERS

License usage for non-managed or non-LSF workload.

DISPLAYS

Terminall display associated with the license feature

PIDS

Process ID associated with the license feature

EXAMPLES

```
% blusers -l
FEATURE   SERVICE_DOMAIN   USER    HOST     NLICS    NTASKS OTHERS   DISPLAYS PIDS
feat1     LanServer        user1   hostA    1        1      0       (/dev/tty) (16326)
```

```
% blusers -J
JOBID    USER       HOST       PROJECT             CLUSTER        START_TIME
553      user1      hostA      p3                  cluster1       Oct  5 15:47:14
RESOURCE         RUSAGE        SERVICE_DOMAIN
p1_f1            1             app_1
```

SEE ALSO

blhosts(1), blinfo(1), blstat(1)

bmgroup

displays information about host groups

SYNOPSIS

bmgroup [**-r**] [**-l**] [**-w**] [*host_group* ...]

bmgroup [**-h** | **-V**]

DESCRIPTION

Displays host groups and host names for each group.

By default, displays information about all host groups. A host partition is also considered a host group.

OPTIONS

-r

Expands host groups recursively. The expanded list contains only host names; it does not contain the names of subgroups. Duplicate names are listed only once.

-l

Displays static and dynamic host group members. A '+' sign before the host name indicates that the host is dynamic and is currently a member of the host group. A '-' sign before the host name indicates that the host is currently not an LSF host but is a member of the dynamic host group.

Also identifies condensed host groups. These host groups are defined by `CONDENSE` in the `HostGroup` section of `lsb.hosts`.

-w

Wide format. Displays host and host group names without truncating fields.

host_group ...

Only displays information about the specified host groups. Do not use quotes when specifying multiple host 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 and host partitions are defined in the configuration file `lsb.hosts`(5).

SEE ALSO

`lsb.hosts`(5), `bugroup`(1), `bhosts`(1)

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 and rerunnable jobs. LSF administrators and `root` can migrate jobs submitted by other users.

By default, migrates one job, 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.

To migrate a job, both hosts must be binary compatible, run the same OS version, have access to the executable, have access to all open files (LSF must locate them with an absolute path name), and have access to the checkpoint directory.

Only started jobs can be migrated (i.e., running or suspended jobs); pending jobs cannot be migrated.

Members of a chunk job can be migrated. Chunk jobs in WAIT state are removed from the job chunk and put into PEND state.

When a checkpointable job is migrated, LSF checkpoints and kills the job (similar to the `-k` option of `bchkpnt(1)`) then restarts it on the next available host. If checkpoint is not successful, the job is not killed and remains on the host. If a job is being checkpointed when `bmig` is issued, the migration is ignored. This situation may occur if periodic checkpointing is enabled.

With the MultiCluster job forwarding model, you can only operate on a MultiCluster job from the execution cluster, and the job will be restarted on the same host. To move the job to a different host, use `brun`. Use `brun -b` if another host might not have access to the checkpoint directory.

When a rerunnable job is migrated, LSF kills the job (similar to `bkill`(1)) then restarts it from the beginning on the next available host.

The environment variable LSB_RESTART is set to Y when a migrating job is restarted or rerun.

A job is made rerunnable by specifying the `-r` option on the command line using `bsub`(1) and `bmod`(1), or automatically by configuring RERUNNABLE in `lsb.queues`(5).

A job is made checkpointable by specifying the location of a checkpoint directory on the command line using the `-k` option of `bsub`(1) and `bmod`(1), or automatically by configuring CHKPNT in `lsb.queues`(5).

OPTIONS

-f

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

job_ID | **"***job_ID***[***index_list***]"** | **0**

Specifies the job ID of the jobs to be migrated. The -J and -u options are ignored.

If you specify a job ID of **0** (zero), all other job IDs are ignored, and all jobs that satisfy the -J and -u options are migrated.

If you do not specify a job ID, the most recently submitted job that satisfies the -J and -u options is migrated.

In a MultiCluster environment, use the local job ID.

-J *job_name*

Specifies the job name of the job to be migrated. Ignored if a job ID other than 0 (zero) is specified.

-m "*host_name* ..." | **-m** **"***host_group* ...**"**

Migrate the jobs to the specifed hosts.

This option cannot be used on a MultiCluster job.

-u **"***user_name***"** | **-u** **"***user_group***"** | **-u all**

Specifies that only jobs submitted by these users are to be migrated.

If the reserved user name all is specified, jobs submitted by all users are to be migrated. 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(1), brestart(1), bchkpnt(1), bjobs(1), bqueues(1), bhosts(1), bugroup(1), mbatchd(8), lsb.queues(5), kill(1)

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 [**-sla** *service_class_name* | **-slan**] [*job_ID*]

bmod [**-h** | **-V**]

OPTION LIST

[**-B** | **-Bn**]
[**-N** | **-Nn**]
[**-r** | **-rn**]
[**-x** | **-xn**]
[**-a** *esub_parameters* | **-an**]
[**-b** *begin_time* | **-bn**]
[**-C** *core_limit* | **-Cn**]
[**-c** [*hour***:**]*minute*[**/***host_name* | **/***host_model*] | **-cn**]
[**-D** *data_limit* | **-Dn**]
[**-e** *err_file* | **-en**]
[**-E** **"***pre_exec_command* [*argument* ...]**"** | **-En**]
[**-eo** *err_file* | **-en**]
[**-ext**[**sched**] **"***external_scheduler_options***"**]
[**-f** **"***local_file op* [*remote_file*]**"** ... | **-fn**]
[**-F** *file_limit* | **-Fn**]
[**-g** *job_group_name* | **-gn**]
[**-G** *user_group* | **-Gn**]
[**-i** *input_file* | **-in** | **-is** *input_file* | **-isn**]
[**-J** *job_name* | **-J** **"%***job_limit*" | **-Jn**]
[**-k** *checkpoint_dir* | **-k** **"***checkpoint_dir* [*checkpoint_period*]
[**method=***method_name*]**"** | **-kn**]
[**-L** *login_shell* | **-Ln**]
[**-Lp** *ls_project_name* | **-Lpn**][*job_ID* | "*job_ID*[*index_list*]"]
[**-a** *additional esub info*]
[**-m** **"***host_name*[**@***cluster_name*][**+**[*pref_level*]] | *host_group*[**+**[*pref_level*]] ...**"** |**-mn**]
[**-M** *mem_limit* | **-Mn**]
[**-n** *num_processors* | **-nn**]
[**-o** *out_file* | **-on**]
[**-oo** *out_file* | **-on**]
[**-P** *project_name* | **-Pn**]
[**-q** "*queue_name* ..." | **-qn**]
[**-R** "*res_req*" | **-Rn**]
[**-s** *signal* | **-sn**]
[**-S** *stack_limit* | **-Sn**]
[**-sla** *service_class_name* | **-slan**]
[**-sp** *priority* | **-spn**]

[**-t** *term_time* | **-tn**]
[**-U** *reservation_ID* | **-Un**]
[**-u** *mail_user* | **-un**]
[**-w '***dependency_expression***'** | **-wn**]
[**-wa '**[*signal* | *command* | **CHKPNT**]**'** | **-wan**]
[**-wt '***job_warning_time***'** | **-wtn**]
[**-W** *run_limit* [/*host_name* | /*host_model*] | **-Wn**]
[**-Z "***new_command***"** | **-Zs "***new_command***"** | **-Zsn**]
[*job_ID* | **"***job_ID***[***index***]"**]

DESCRIPTION

Modifies the options of a previously submitted job. See `bsub(1)` for complete descriptions of job submission options you can modify with `bmod`.

Only the owner of the job, or LSF administrators, can modify the options of a job.

All options specified at submission time may be changed. The value for each option may be overridden with a new value by specifying the option as in `bsub`. To reset an option to its default value, use the option string followed by '`n`'. Do not specify an option value when resetting an option.

The `-i`, `-in`, and `-Z` options have counterparts that support spooling of input and job command files (`-is`, `-isn`, `-Zs`, and `-Zsn`).

You can modify all options of a pending job, even if the corresponding `bsub` option was not specified.

Modifying a job that is pending in a chunk job queue (CHUNK_JOB_SIZE) removes the job from the chunk to be scheduled later.

Like `bsub`, `bmod` also calls `mesub` and any existing `esub` executables. `bmod` cannot make changes to the job environment through `esub`. Environment changes only occur when `esub` is called by the original job submission with `bsub`.

Modifying running jobs

By default, you can modify resource requirements for running jobs (**-R "***res_req***"**). To modify additional job options for running jobs, define LSB_MOD_ALL_JOBS=Y in `lsf.conf`.

When LSB_MOD_ALL_JOBS=Y is set, the following are the only `bmod` options that are valid for running jobs. You cannot make any other modifications after a job has been dispatched.

- CPU limit (**-c** [*hour***:**]*minute*[**/***host_name* | **/***host_model*])

- Job group (**-g** *job_group_name*)

- Memory limit (**-M** *mem_limit*)

- Rerunnable jobs (**-r** | **-rn**)

- Resource reservation (**-R "***res_req***"**)

- Run limit (**-W** *run_limit*[**/***host_name* | **/***host_model*])

- Standard output (`stdout`) file name (**-o** *output_file*)

- Standard error (`stderr`) file name (**-e** *error_file*)

- Overwrite standard output (`stdout`) file name (**-oo** *output_file*)

- Overwrite standard error (`stderr`) file name (**-eo** *error_file*)

Modified resource usage limits cannot exceed limits 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 `lsf.conf`.

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 will be rejected if one of your previous arrays has the same name but a different index.

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]`.

Modifying advance reservations

If advance reservations are enabled, administrators can use the `-U` option of `bmod` to change a job to another reservation ID. For example:

```
% bmod -U user1#0 1234
```

To cancel the reservation, use the `-Un` option of `bmod`. For example:

```
% bmod -Un 1234
```

Closed reservations

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.

For example:

```
% bmod -t 15:0 1234
```

You can only modify the termination time of an advance reservation job while the reservation is active. When the reservation expires or is deleted, the job is killed and you can no longer make any modifications.

Open reservations

LSF users can modify the termination time of their open advance reservations without interfering with current job scheduling policies. For example, you specify two open advance reservations as follows:

```
% brsvadd -o -n 1 -m hostA -u user1 -b 15:10 -e 15:30
Reservation user1#17 is created
% brsvadd -o -n 1 -m hostA -u user1 -b 15:35 -e 15:50
Reservation user1#18 is created
```

At 15:15, while job 122 in `user1#17` is running, you modify its termination time as follows:

```
% bmod -t 15:40 122
```

This termination time overlaps the reservation window of `user1#18`, in which job 245 started at 15:35. After modifying the termination time of job 122, the following events occur:

At this time ...	These events occur ...
15:35	◆ Job 122 in reservation user1#17 suspends ◆ Job 245 in reservation user1#18 starts
15:40	◆ Job 122 in reservation user#17 is still suspended ◆ Job 245 in reservation user#18 continues to run
15:50	◆ Job 122 in reservation user1#17 suspends ◆ Job 245 in reservation user1#18 suspends
15:51	◆ Job 122 in reservation user1#17 runs ◆ Job 245 in reservation user1#18 suspends

At 15:50, both Job 122 in reservation `user#17` and Job 245 in reservation `user#18` are suspended. At this point, the existing scheduling policies determine which job runs; in this example, Job 122 in reservation `user#17` runs.

bmod -t will not change the termination time of a pending job.

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:

```
% bmod -g /risk_group/portfolio2/monthly 105
```

moves job 105 to job group `/risk_group/portfolio2/monthly`.

Like `bsub -g`, if the job group does not exist, LSF creates it.

`bmod -g` cannot be combined with other `bmod` options. It can operate on finished, running, and pending 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.

You cannot modify the job group of a job attached to a service class.

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:

```
% bmod -sla Kyuquot 2307
```

attaches job 2307 to the service class `Kyuquot`.

```
% bmod -slan 2307
```

detaches job 2307 from the service class `Kyuquot`.

You cannot:

- Use -sla with other bmod options
- Move job array elements from one service class to another, only entire job arrays
- Modify the service class of job already attached to a job group

Use bsla to display the properties of service classes configured in LSB_CONFDIR/*cluster_name*/configdir/lsb.serviceclasses (see lsb.serviceclasses(5))and dynamic information about the state of each configured service class.

OPTIONS

job_ID | **"***job_ID***[***index***]"**

Modifies jobs with the specified job ID.

Modifies job array elements specified by **"***job_ID***[***index***]"**.

-h

Prints command usage to stderr and exits.

-V

Prints LSF release version to stderr and exits.

LIMITATIONS

Modifying remote running jobs in a MultiCluster environment is not supported.

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(1)

bparams

displays information about configurable system parameters in `lsb.params`

SYNOPSIS

bparams [**-l**]

bparams [**-h** | **-V**]

DESCRIPTION

By default, displays only the interesting parameters.

OPTIONS

-l

Long format. Displays detailed information about all the configurable parameters in `lsb.params`.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

SEE ALSO

`lsb.params`(5)

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 have been produced by one of your unfinished jobs, up to the time that this command is invoked.

By default, displays the output 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 terminating an erroneous job.

OPTIONS

-f

Displays the output of the job using the command `tail -f`.

-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 has been dispatched to the specified host.

-J *job_name*

Operates on your most recently submitted job that has the specified job name.

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

`tail`(1), `bsub`(1), `bjobs`(1), `bhist`(1), `bhosts`(1), `bqueues`(1)

bpost

sends external status messages and attaches data files to a job

SYNOPSIS

bpost [**-i** *message_index*] [**-d** **"***description***"**] [**-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. Done or exited jobs cannot accept messages.

By default, operates on the message index 0. By default, posts the message `"no description"`.

If a you specify a job ID:

- You can only send messages and data 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 `-J` *job_name*:

- You can ony send messages and data to your own jobs.
- You cannot send messages and data to jobs submitted by other users.
- Root and the LSF administrators can only send messages and data 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 `bpost`(1) to attach a data file to the job, then use the `bread`(1) 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 `bread`(1) or `bstatus`(1) to retrieve the messages posted to the job. Use `bread`(1) to copy message attachments to external files.

For example, your application may require additional job status descriptions besides the ones that LSF provides internally (PEND, RUN, SUSP, etc.) 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` to update the external job status. The command:

```
$ bstatus -d "description" myjob
```

is equivalent to:

```
$ bpost -i 0 -d "description" myjob
```

With MultiCluster, both clusters must run LSF Version 6.2 or later. You cannot post a message to a MultiCluster job if the clusters are disconnected. You cannot attach files to MultiCluster jobs.

OPTIONS

-i *message_index*

Operates on the specified message index.

Default: 0

Use the MAX_JOB_MSG_NUM parameter in `lsb.params` to set a maximum number of messages for a job. With MultiCluster, to avoid conflicts, MAX_JOB_MSG_NUM should be the same in all clusters.

-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 may require additional job status descriptions besides the ones that LSF provides internally (`PEND`, `RUN`, `SUSP`, etc.)

Default: `"no description"`

-a *data_file*

Attaches the specified data file to the job external storage. This option is ignored for MultiCluster jobs; you can only attach a file if the job executes in the local cluster.

Use the JOB_ATTA_DIR parameter in `lsb.params`(5) to specify the directory where attachment data files are saved. The directory must have at least 1 MB of free space. `mbatchd` checks for available space in the job attachment directory before transferring the file.

Use the MAX_JOB_ATTA_SIZE parameter in `lsb.params` to set a maximum size for job message attachments.

job_ID | **"***job_ID***[***index***]"** | **-J** *job_name*

Required. Operates on the specified job. With MultiCluster job forwarding model, you must always use the local job ID.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

EXAMPLE

```
% bpost -i 1 -d "step 1" -a step1.out 2500
```

Puts the message text `step 1` into message index 1, and attaches the file `step1.out` to job 2500.

SEE ALSO

`bread`(1), `bstatus`(1), MAX_JOB_ATTA_SIZE, MAX_JOB_MSG_NUM

bqueues

displays information about queues

SYNOPSIS

bqueues [**-w** | **-l** | **-r**] [**-m** *host_name* | **-m** *host_group* | **-m** *cluster_name* | **-m all**]
[**-u** *user_name* | **-u** *user_group* | **-u all**] [*queue_name* ...]

bqueues [**-h** | **-V**]

DESCRIPTION

Displays information about queues.

By default, returns the following information about all queues: queue name, queue priority, queue status, job slot statistics, and job state statistics.

In MultiCluster, returns the information about all queues in the local cluster.

Batch queue names and characteristics are set up by the LSF administrator (see `lsb.queues`(5) and `mbatchd(8)`).

CPU time is normalized.

OPTIONS

-w

Displays queue information in a wide format. Fields are displayed without truncation.

-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, and job controls.

Also displays user shares.

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.

-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.

-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 see `bmgroup(1)`.

In MultiCluster, 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 see `bugroup(1)`).

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.

OUTPUT

Default Output

Displays the following fields:

QUEUE_NAME

The name of the queue. Queues are named to correspond to the type of jobs 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 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 will not be started unless they are switched to other queues (see `bswitch`).

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, suspension and resumption time. Jobs from higher priority queues are dispatched first (this is contrary to UNIX process priority ordering), and jobs from lower priority queues are suspended first when hosts are overloaded.

STATUS

The current status of the queue. The possible values are:

Open

The queue is able to accept jobs.

Closed

The queue is not able to accept jobs.

Active

Jobs in the queue may be started.

Inactive

Jobs in the queue cannot be started for the time being.

At any moment, each queue is either `Open` or `Closed`, and is either `Active` or `Inactive`. The queue can be opened, closed, inactivated and re-activated by the LSF administrator using `badmin` (see `badmin(8)`).

Jobs submitted to a queue that is later closed are still dispatched as long as the queue is active. The queue can also become inactive when either its dispatch window is closed or its run window is closed (see DISPATCH_WINDOWS in the "Output for the -l Option" section). In this case, the queue cannot be activated using `badmin`. The queue is re-activated 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 boot time is set to open, and either active or inactive depending on its 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 which have not yet finished, and by pending jobs which have slots reserved for them.

A sequential job will use one job slot when it is dispatched to a host, while a parallel job will use as many job slots as is required by `bsub -n` when it is dispatched. See `bsub(1)` for details. If '–' is displayed, there is 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 which have not yet finished, and by pending jobs which have slots reserved for them. If '–' is displayed, there is no limit.

JL/P

The maximum number of job slots a processor can process from the queue. This includes job slots of dispatched jobs that have not yet finished, and job slots reserved for some pending jobs. The job slot limit per processor (JL/P) controls the number of jobs sent to each host. This limit is configured per processor so that multiprocessor hosts are automatically allowed to run more jobs. If '–' is displayed, there is no limit.

JL/H

The maximum number of job slots a host can allocate from this queue. This includes the job slots of dispatched jobs that have not yet finished, and those reserved for some pending jobs. The job slot limit per host (JL/H) controls the number of jobs sent to each host, regardless of whether a host is a uniprocessor host or a multiprocessor host. If '–' is displayed, there is no limit.

NJOBS

The total number of job slots held currently by jobs in the queue. 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 queue. See `bjobs(1)` for an explanation of batch job states.

PEND

The number of job slots used by pending jobs in the queue.

RUN

The number of job slots used by running jobs in the queue.

SUSP

The number of job slots used by suspended jobs in the queue.

Long Output (-l)

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

Description

A description of the typical use of the queue.

Default queue indication

Indicates that this is the default queue.

PARAMETERS/STATISTICS

NICE

The nice value at which jobs in the queue will be run. This is the UNIX nice value for reducing the process priority (see `nice(1)`).

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 has been inactivated by the LSF administrator.

SSUSP

The number of job slots in the queue allocated to jobs that are suspended by LSF because of load levels or run windows.

USUSP

The number of job slots in the queue allocated to jobs that are suspended by the job submitter or by the LSF administrator.

RSV

The number of job slots in the queue that are reserved by LSF for pending jobs.

Migration threshold

The length of time in seconds that a job dispatched from the queue will remain suspended by the system before LSF attempts to migrate the job to another host. See the MIG parameter in `lsb.queues` and `lsb.hosts`.

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 `lsb.queues`.

Interval for a host to accept two jobs

The length of time in seconds to wait after dispatching a job to a host before dispatching a second job to the same host. If the job accept interval is zero, a host may accept more than one job in each dispatching interval. See the JOB_ACCEPT_INTERVAL parameter in `lsb.queues` and `lsb.params`.

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 limits are:

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 belonging 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 to the job to kill it.

PROCLIMIT

The maximum number of processors allocated to a job. Jobs that request fewer slots than the minimum PROCLIMIT or more slots than the maximum PROCLIMIT are rejected. If the job requests minimum and maximum job slots, the maximum slots requested cannot be less than the minimum PROCLIMIT, and the minimum slots requested cannot be more than the maximum PROCLIMIT.

MEMLIMIT

The maximum running set size (RSS) of a process, in KB. If a process uses more than MEMLIMIT kilobytes of memory, 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.

SWAPLIMIT

The swap space limit that a job may use. If SWAPLIMIT is reached, the system sends the following signals in sequence to all processes in the job: `SIGINT`, `SIGTERM`, and `SIGKILL`.

PROCESSLIMIT

The maximum number of concurrent processes allocated to a job. If PROCESSLIMIT is reached, the system sends the following signals in sequence to all processes belonging to the job: `SIGINT`, `SIGTERM`, and `SIGKILL`.

THREADLIMIT

The maximum number of concurrent threads allocated to a job. If THREADLIMIT is reached, the system sends the following signals in sequence to all processes belonging to the job: SIGINT, SIGTERM, and SIGKILL.

The possible UNIX per-process resource limits are:

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 has been in the 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 kilobytes. 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 kilobytes. This 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, in kilobytes. This 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.

CORELIMIT

The maximum size of a core file, in KB. This limit is enforced by the setrlimit system call if it supports the RLIMIT_CORE option.

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.

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 defined for hosts (see bhosts(1) and lsb.hosts(5)). 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 utilization 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 kilobytes per second.

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in /tmp, in megabytes.

swp

The amount of currently available swap space, in megabytes.

mem

The amount of currently available memory, in megabytes.

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 dispatching thresholds for the corresponding load indices. If '–' is displayed as the value, it means the threshold is not applicable. Jobs in the queue may be dispatched to a host if the values of all the load indices of the host are within (below or above, depending on the meaning of the load index) the corresponding thresholds of the queue and the host. The same conditions are used to resume jobs dispatched from the queue that have been 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 will be 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 have 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 have 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 have 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 may be configured:

FAIRSHARE

Queue-level fairshare scheduling is enabled. Jobs in this queue are scheduled based on a fairshare policy instead of the first-come, first-serve (FCFS) policy.

BACKFILL

A job in a backfill queue can use the slots 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 are reached. Backfilling is not supported when MAX_JOBS, QJOB_LIMIT, and UJOB_LIMIT are reached.

IGNORE_DEADLINE

If IGNORE_DEADLINE is set to Y, starts all jobs regardless of the run limit.

EXCLUSIVE

Jobs dispatched from an exclusive queue can run exclusively on a host if the user so specifies at job submission time (see `bsub(1)`). Exclusive execution means that the job is sent to a host with no other batch job running there, and no further job, batch or interactive, will be 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. (see the `-I`, `-Is`, and `-Ip` options of `bsub(1)`). The default is to accept both interactive and non-interactive jobs.

ONLY_INTERACTIVE

This queue only accepts batch interactive jobs. Jobs must be submitted using the `-I`, `-Is`, and `-Ip` options of `bsub(1)`. The default is to accept both interactive and non-interactive jobs.

FAIRSHARE_QUEUES

Lists queues participating in cross-queue fairshare. The first queue listed is the master queue—the queue in which fairshare is configured; all other queues listed inherit the fairshare policy from the master queue. Fairshare information applies to all the jobs running in all the queues in the master-slave set.

DISPATCH_ORDER

DISPATCH_ORDER=QUEUE 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. This avoids having users with higher fairshare priority getting jobs dispatched from low-priority queues.

USER_SHARES

A list of [*user_name*, *share*] pairs. *user_name* is either a user name or a user group name. *share* is the number of shares of resources assigned to the user or user group. A party will get a portion of the resources proportional to that party's share divided by the sum of the shares of all parties specified in this queue.

DEFAULT HOST SPECIFICATION

The default host or host model that will be used to normalize the CPU time limit of all jobs.

If you want to view a list of the CPU factors defined for the hosts in your cluster, see `lsinfo(1)`. The CPU factors are configured in `lsf.shared(5)`.

The appropriate CPU scaling factor of the host or host model is used to adjust the actual CPU time limit at the execution host (see CPULIMIT in `lsb.queues(5)`). The DEFAULT_HOST_SPEC parameter in `lsb.queues` overrides the system DEFAULT_HOST_SPEC parameter in `lsb.params` (see `lsb.params(5)`). If a user explicitly gives a host specification when submitting a job using `bsub -c` *cpu_limit*[/*host_name* | /*host_model*], the user specification overrides the values defined in both `lsb.params` and `lsb.queues`.

RUN_WINDOWS

The time windows in a week during which jobs in the queue may run.

When a queue is out of its window or windows, no job in this queue will be 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 may be dispatched.

When a queue is out of its dispatch window or windows, no job in this queue will be dispatched. Jobs 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). Note that such windows are only applicable to batch jobs. Interactive jobs scheduled by LIM are controlled by another set of dispatch windows (see `lshosts(1)`). Similar dispatch windows may be configured for individual hosts (see `bhosts(1)`).

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 '–', 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 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. See `bugroup(1)`.

If the fairshare scheduling policy is enabled, users cannot submit jobs to the queue unless they also have a share assignment. This also applies to LSF administrators.

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. See `bmgroup(1)`.

NQS DESTINATION QUEUES

A list of NQS destination queues to which this queue can dispatch jobs.

When you submit a job using `bsub -q` *queue_name*, 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 running on an NQS system outside the LSF cluster, the job is still managed by LSF in almost the same way as jobs running inside the LSF cluster. Thus, you may have your batch jobs transparently sent to an NQS system to run and then get the results of your jobs back. You may use any supported user interface, including LSF commands and NQS commands (see `lsnqs(1)`) to submit, monitor, signal and delete your batch jobs that are running in an NQS system. See `lsb.queues(5)` and `bsub(1)` for more information.

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. See `lsb.queues(5)` for more information.

PRE_EXEC

The queue's pre-execution command. The pre-execution command is executed before each job in the queue is run on the execution host (or on the first host selected for a parallel batch job). See `lsb.queues(5)` for more information.

POST_EXEC

The queue's post-execution command. The post-execution command is run on the execution host when a job terminates. See `lsb.queues(5)` for more information.

REQUEUE_EXIT_VALUES

Jobs that exit with these values are automatically requeued. See `lsb.queues(5)` for more information.

RES_REQ

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

Maximum slot reservation time

The maximum time in seconds a slot is reserved for a pending job in the queue. See the SLOT_RESERVE=MAX_RESERVE_TIME[n] parameter in `lsb.queues`.

RESUME_COND

The conditions that must be satisfied to resume a suspended job on a host. See `lsb.queues(5)` for more information.

STOP_COND

The conditions which determine whether a job running on a host should be suspended. See `lsb.queues(5)` for more information.

JOB_STARTER

An executable file that runs immediately prior to the batch job, taking the batch job file as an input argument. All jobs submitted to the queue are run via the job starter, which is generally used to create a specific execution environment before processing the jobs themselves. See `lsb.queues(5)` for more information.

CHUNK_JOB_SIZE

Chunk jobs only. Specifies the maximum number of jobs 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 takes 1 to 2 minutes to run.

SEND_JOBS_TO

MultiCluster. List of remote queue names to which the queue forwards jobs.

RECEIVE_JOBS_FROM

MultiCluster. List of remote cluster names from which the queue receives jobs.

PREEMPTION

PREEMPTIVE

The queue is preemptive. Jobs in a preemptive queue may 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 a preemptable queue may be preempted by jobs in higher-priority queues, even if the higher-priority queues are not specified as preemptive.

RERUNNABLE

If the RERUNNABLE field displays `yes`, jobs in the queue are rerunnable. That is, jobs in the queue are automatically restarted or rerun if the execution host becomes unavailable. However, a job in the queue will not be restarted if the you have removed the rerunnable option from the job. See `lsb.queues(5)` for more information.

CHECKPOINT

If the CHKPNTDIR field is displayed, jobs in the queue are checkpointable. Jobs will use the default checkpoint directory and period unless you specify other values. Note that a job in the queue will not be checkpointed if you have removed the checkpoint option from the job. See `lsb.queues(5)` for more information.

CHKPNTDIR

Specifies the checkpoint directory using an absolute or relative path name.

CHKPNTPERIOD

Specifies the checkpoint period in seconds.

Although the output of `bqueues` reports the checkpoint period in seconds, the checkpoint period is defined in minutes (the checkpoint period is defined through the `bsub -k "`*checkpoint_dir* [*checkpoint_period*]`"` option, or in `lsb.queues`).

JOB CONTROLS

The configured actions for job control. See JOB_CONTROLS parameter in `lsb.queues`.

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 the queue control commands `qclose`, `qopen`, `qact`, and `qinact`, `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. SLOT_SHARE 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 only belong to one pool. All queues in the pool must share the same set of hosts.

Recursive Share Tree Output (-r)

In addition to the fields displayed for the `-l` option, the `-r` option displays the following:

SCHEDULING POLICIES

FAIRSHARE

The `-r` option causes `bqueues` to recursively display the entire share information tree associated with the queue.

SEE ALSO

`bugroup`(1), `nice`(1), `getrlimit`(2), `lsb.queues`(5), `bsub`(1), `bjobs`(1), `bhosts`(1), `badmin`(8), `mbatchd`(8)

bread

reads messages and attached data files from a job

SYNOPSIS

bread [**-i** *message_index*] [**-a** *file_name*]
job_ID | **"***job_ID***[***index***]"** | **-J** *job_name*

bread [**-h** | **-V**]

DESCRIPTION

Reads messages and data posted to an unfinished job with `bpost`.

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 a you specify a job ID:

- You can get read messages of jobs submitted by other users, but you cannot read data files attached to jobs submitted by other users.
- You can only read data files attached to your own jobs.
- Root and LSF administrators can read messages of jobs submitted by other users.
- Root and LSF administrators cannot read data files attached to jobs submitted by other users.

Job names are not unique; if you specify `-J` *job_name*:

- You only can read messages and data from your own jobs.
- You cannot read messages and data from jobs submitted by other users.
- Root and the LSF administrators can only read messages and data from their own jobs.

The command:

```
% bstatus
```

is equivalent to:

```
% bread -i 0
```

OPTIONS

-i *message_index*

Specifies the message index to be retrieved.

Default: 0

-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 MultiCluster 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 already exists, `-a` overwrites it with the new file.

By default, `-a` gets the attachment file from the directory specified by the JOB_ATTA_DIR parameter. If JOB_ATTA_DIR is not specified, job message attachments are saved in `LSB_SHAREDIR/info/`.

job_ID* | "*job_ID*[*index*]" | -J *job_name

Required. Specify the job to operate on.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

EXAMPLE

```
% bpost -i 1 -d "step 1" -a step1.out 2500
% bread -i 1 -a step2.in 2500
JOBID      MSG_ID FROM      POST_TIME      DESCRIPTION
2500       1      user1     May 19 13:59   step 1
```

Displays the message description text `step 1` for message index 1 of job 2500 and copies the data in the file `step1.out` attached to message 1 to the file `step2.in`.

SEE ALSO

`bpost`(1), `bstatus`(1), `bsub`(1), `JOB_ATTA_DIR`

brequeue

Kills and requeues a job

SYNOPSIS

brequeue **[-J** *job_name* | **-J "***job_name***[***index_list***]"]** [**-u** *user_name* | **-u all**] [*job_ID* | **"***job_ID***[***index_list***]"**] [**-d**] [**-e**] [**-r**] [**-a**] [**-H**]

brequeue [**-h** | **-V**]

DESCRIPTION

You can only use `brequeue` on a job you own, unless you are `root` or the LSF administrator.

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. Once dispatched, the job starts over from the beginning. The requeued job keeps the same job ID.

Use brequeue to requeue job arrays or elements of them.

By default, kills and requeues your most recently submitted job when no job_ID is specified.

With MultiCluster, you can only use brequeue on jobs in local queues. A job that is killed and requeued is assigned a new job ID on the cluster in which it is executed, but it retains the same job ID on the cluster from which it was submitted. For example, a job from cluster A that is killed and requeued and then run on cluster B will be assigned a new job ID on cluster B. However, when the `bjobs` command is used from cluster A, the submitting cluster, the job will be displayed with the original job ID. When the `bjobs` command is used from cluster B, the execution cluster, the job will be displayed with the new job ID.

OPTIONS

-J *job_name* | -J "*job_name*[*index_list*]"

Operates on the specified job.

Since job names are not unique, multiple job arrays may have the same name with a different or same set of indices.

-u *user_name* | -u all

Operates on the specified user's jobs or all jobs.

Only `root` and LSF administrators can requeue jobs 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.

-d

Requeues jobs that have finished running with `DONE` job status.

-e

Requeues jobs that have terminated abnormally with `EXIT` job status.

-r

Requeues jobs that are running.

-a

Requeues all jobs including running jobs, suspending jobs, and jobs with `EXIT` or `DONE` status.

-H

Requeues jobs to `PSUSP` job status.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

LIMITATIONS

`brequeue` cannot be used on interactive batch jobs; `brequeue` only kills interactive batch jobs, it does not restart them.

bresources

displays information about resource reservation and resource limits configuration.

SYNOPSIS

bresources [**-s**] [*resource_name* ...]

bresources [**-h** | **-V**]

DESCRIPTION

By default, `bresources` displays all resource configurations in `lsb.resources`. This is the same as `blimits -c`.

OPTIONS

-s

Displays per-resource reservation configurations from the ReservationUsage section of `lsb.resources`. For example:

```
% bresources -s
Begin ReservationUsage
RESOURCE                  METHOD
licenseX                  PER_JOB
licenseY                  PER_HOST
licenseZ                  PER_SLOT
End ReservationUsage
```

resource_name ...

Only displays information about the specified resource. For example:

```
% bresources -s  licenseZ
RESOURCE                  METHOD
licenseZ                  PER_SLOT
```

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

brestart

restarts checkpointed jobs

SYNOPSIS

brestart [*bsub_options*] [**-f**] *checkpoint_dir* [*job_ID* | **"***job_ID***[***index***]"**]

brestart [**-h** | **-V**]

OPTION LIST

-B
-f
-N
-x
-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* ...**"**
-S *stack_limit*
-t *term_time*
-w **'***dependency_expression***'**
-W *run_limit*[**/***host_name*| **/***host_model*]
checkpoint_dir [*job_ID* | **"***job_ID***[***index***]"**]

DESCRIPTION

Restarts a checkpointed job using the checkpoint files saved in *checkpoint_dir/last_job_ID/*. Only jobs that have been successfully checkpointed can be restarted.

Jobs are re-submitted and assigned a new job ID. The checkpoint directory is renamed using the new job ID, *checkpoint_dir/new_job_ID/*.

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.

To restart a job on another host, both hosts must be binary compatible, run the same OS version, have access to the executable, have access to all open files (LSF must locate them with an absolute path name), and have access to the checkpoint directory.

The environment variable LSB_RESTART is set to Y when a job is restarted.

LSF invokes the `erestart`(8) executable found in LSF_SERVERDIR to perform the restart.

Only the `bsub` options listed here can be used with `brestart`.

Like bsub, brestart also calls mesub and any existing esub executables. brestart cannot make changes to the job environment through esub. Environment changes only occur when esub is called by the original job submission with bsub.

OPTIONS

Only the bsub options listed in the option list above can be used for brestart. Except for the following option, see bsub(1) for a description of brestart options.

-f

Forces the job to be restarted even if non-restartable conditions exist (these conditions are operating system specific).

LIMITATIONS

In kernel-level checkpointing, you cannot change the value of core limit, CPU limit, stack limit or memory limit with brestart.

SEE ALSO

bsub(1), bjobs(1), bmod(1), bqueues(1), bhosts(1), bchkpnt(1), lsb.queues(5), echkpnt(8), erestart(8), mbatchd(8)

bresume

resumes one or more suspended jobs

SYNOPSIS

bresume [**-g** *job_group_name*] [**-J** *job_name*] [**-m** *host_name*] [**-q** *queue_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 submitted by other users. You cannot resume a job that is not suspended. Using `bresume` on a job that is not in either the PSUSP or the USUSP state has no effect.

You must specify a job ID or `-g`, `-J`, `-m`, `-u`, or `-q`. You cannot resume a job that is not suspended. Specify `-0` (zero) to resume multiple jobs.

You can also use `bkill -s CONT` to send the resume signal to a job.

If a signal request fails to reach the job execution host, LSF will retry the operation later when the host becomes reachable. LSF retries the most recent signal request.

Jobs that are suspended by the administrator can only be resumed by the administrator or `root`; users do not have permission to resume a job suspended by another user or the administrator. Administrators or `root` can resume jobs suspended by users or administrators.

ENABLE_USER_RESUME parameter (lsb.params)

If ENABLE_USER_RESUME=Y in `lsb.params`, users can resume their own jobs that have been suspended by the administrator.

OPTIONS

0

Resumes all the jobs that satisfy other options (`-g`, `-m`, `-q`, `-u`, and `-J`).

-g *job_group_name*

Resumes only jobs in the job group specified by *job_group_name*.

-J *job_name*

Resumes only jobs with the specified name.

-m *host_name*

Resumes only jobs dispatched to the specified host.

-q *queue_name*

Resumes only jobs in the specified queue.

-u *user_name* | **-u** *user_group* | **-u all**

Resumes only jobs owned by the specified user or group, or all users if the reserved user name `all` is specified.

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

Resumes 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

```
% bresume -q night 0
```

Resumes all of the user's suspended jobs that are 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`(1), `bjobs`(1), `bqueues`(1), `bhosts`(1), `bstop`(1), `bkill`(1), `bgadd`(1), `bgdel`(1), `bjgroup`(1), `bparams`(5), `mbatchd`(8), `kill`(1), `signal`(2) `lsb.params`(5)

brlainfo

displays host topology information

SYNOPSIS

brlainfo [**-l**] [*host_name* ...]

brlainfo [**-h** | **-V**]

DESCRIPTION

`brlainfo` contacts the Platform LSF HPC topology adapter (RLA) on the specified host and presents topology information to the user. By default, displays information about all hosts running RLA.

OPTIONS

-l

Long format. Displays additional host topology information. See “OUTPUT” for a description of information that is displayed.

host_name ...

Only displays information about the specified host.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

Default output

Displays the following fields:

HOSTNAME

Name of the host running 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 additional fields:

FREE CPU LIST

List of free CPUs in the cpuset

For example:

`0-2`

NFREECPUS ON EACH NODE

Number of free CPUs on each node

For example:

`2/0,1/1`

STATIC CPUSETS

List of static cpuset names

For example:

`NO STATIC CPUSETS`

CPU_RADIUS

Available CPUs with a given radius. CPU radius is determined by the processor topology of the system and is expressed in terms of the number of router hops between CPUs. The CPU radius is displayed as a comma-separated list of the number of free CPUs available with radius 0, radius 1, radius 2, and so on:

For example:

`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 therefore have better communications performance.

EXAMPLES

```
% brlainfo hostA hostB hostC
HOSTNAME            CPUSET_OS   NCPUS   NFREECPUS NNODES   NCPU/NODE NSTATIC_CPUSETS
hostA               SGI_IRIX    2       2         1        2         0
hostB               PROPACK_4   4       4         2        2         0
hostC               PROPACK_4   4       3         2        2         0
```

```
% brlainfo -l
HOST: hostC
CPUSET_OS    NCPUS   NFREECPUS NNODES   NCPU/NODE NSTATIC_CPUSETS
PROPACK_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
```

brsvadd

adds an advance reservation

SYNOPSIS

brsvadd [**-o**] **-n** *processors* | **-s** [**-n** *processors*] **-m** **"***host_name* | *host_group* ...**"**
[**-R** **"***res_req***"**] [**-u** *user_name* | **-g** *group_name*] **-b** *begin_time* **-e** *end_time*

brsvadd [**-o**] **-n** *processors* | **-s** [**-n** *processors*] **-m** **"***host_name* | *host_group* ...**"**
[**-R** **"***res_req***"**] [**-u** *user_name* | **-g** *group_name*] **-t** *time_window*

brsvadd [**-o**] **-n** *processors* | **-s** [**-n** *processors*] **-R** **"***res_req***"**
[**-u** *user_name* | **-g** *group_name*] **-b** *begin_time* **-e** *end_time*

brsvadd [**-o**] **-n** *processors* | **-s** [**-n** *processors*] **-R** **"***res_req***"**
[**-u** *user_name* | **-g** *group_name*] **-t** *time_window*

brsvadd [**-h** | **-V**]

DESCRIPTION

By default, this command can only be used by LSF administrators or root.

Reserves processors in advance for a specified period of time for a user or user group, or for system maintenance purposes. Use `-b` and `-e` for one-time reservations, and `-t` for recurring reservations.

To allow users to create their own advance reservations without administrator intervention, configure advance reservation policies in the ResourceReservation section of `lsb.resources`.

Only administrators, root, or the users listed in the ResourceReservation section can add reservations for themselves or any other user or user group.

OPTIONS

-b *begin_time*

Begin time for a one-time reservation. The begin time is in the form

[[[*year*:]*month*:]*day*:]*hour*:*minute*

with 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 `-b` must use the same syntax as the time value for `-e`. It must be earlier than the time value for `-e`, and cannot be earlier than the current time.

-e *end_time*

End time for a one-time reservation. The end time is in the form

`[[[`*`year`*`:]`*`month`*`:]`*`day`*`:]`*`hour`*`:`*`minute`*

with 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 `-e` must use the same syntax as the time value for `-b`. It must be later than the time value for `-b`.

-g *group_name*

Creates a reservation for a user group.

The `-g` *group_name* option does not support the `@cluster` notation for advance reservations on remote clusters.

-m **"***host_name* | *host_group ...***"**

List of hosts for which processors specified with `-n` are reserved. At job submission, LSF considers the hosts in the specified order.

If you also specify a resource requirement string with the `-R` option, `-m` is optional.

The hosts can be local to the cluster or hosts leased from remote clusters.

-n *processors*

Number of processors to reserve. *processors* must be less than or equal to the actual number of CPUs for the hosts selected by `-m` or `-R` for the reservation.

If you also specify the reservation for system use with the `-s` option, `-n` is optional.

-o

Creates an open advance reservation. A job with an open advance reservation will only have the advance reservation property during the reservation window, after which the job becomes a normal job, not subject to termination when the reservation window closes.

This 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.

-R "*res_req*"

Selects hosts for the reservation according to the specified resource requirements. Only hosts that satisfy the resource requirement expression are reserved. `-R` accepts any valid resource requirement string, but only the select string takes effect.

If you also specify a host list with the `-m` option, `-R` is optional.

For more information about resource requirements, see `lsfintro(1)`.

The size of the resource requirement string is limited to 512 bytes.

-s

Creates a reservation for system use. LSF does not dispatch jobs to the specified hosts while the reservation is active.

When specifying a system reservation with `-s`, you do not need to specify the number of processors to reserve with the `-n` option.

-t *time_window*

Time window for a recurring reservation.

The day and time are in the form:

`[`*day*`:]`*hour*`[:`*minute*`]`

with the following ranges:

- *day of the week*: 0-6
- *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*

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.

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.

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.

-u *user_name*

Creates a reservation for an individual user.

The `-u` *user_name* 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 creates a one-time advance reservation for 1024 processors on host `hostA` for user `user1` between 6:00 a.m. and 8:00 a.m. today:

  ```
  % brsvadd -n 1024 -m hostA -u user1 -b 6:0 -e 8:0
  Reservation "user1#0" is created
  ```

  The hosts specified by `-m` can be local to the cluster or hosts leased from remote clusters.
- The following command creates an advance reservation for 1024 processors on two hosts `hostA` and `hostB` for user group `groupA` every Wednesday from 12:00 midnight to 3:00 a.m.:

  ```
  % brsvadd -n 2048 -m "hostA hostB" -g groupA -t "3:0:0-
  3:3:0"
  Reservation "groupA#0" is created
  ```
- The following command creates an open advance reservation for 1024 processors on host hostA for user user1 between 6:00 a.m. and 8:00 a.m. today.

  ```
  % brsvadd -o -n 1024 -m hostA -u user1 -b 6:0 -e 8:0

  Reservation "user1#0" is created
  ```

SEE ALSO

`brsvs`(1), `brsvdel`(8), `lsb.resources`(5)

brsvdel

deletes an advance reservation

SYNOPSIS

brsvdel *reservation_ID ...*

brsvdel [**-h** | **-V**]

DESCRIPTION

By default, this command can only be used by LSF administrators or root.

Deletes advance reservations for the specified reservation IDs.

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 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 `lsb.resources`.

Administrators and root can delete any reservations. Users listed in the ResourceReservation section can only delete reservations they created themselves.

OPTIONS

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

SEE ALSO

`brsvadd`(1), `brsvs`(1), `lsb.resources`(5)

brsvs

displays advance reservations

SYNOPSIS

brsvs [**-l**] [**-p all** | **"***host_name* ...**"**] [**-w**]

brsvs [**-l**] [**-c all** | **"***policy_name***"**] [**-w**]

brsvs [**-h** | **-V**]

DESCRIPTION

By default, displays the current advance reservations for all hosts, users, and groups.

For advance reservations across clusters:

- `-p all` shows local and all remote reservations
- The default `all` includes both local and remote
- *host_name* does NOT take *host_name@cluster_name*

By default, `brsvs` truncates the reservation ID (RSVID) at 11 characters. Use `-w` 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 the reservation type (open or closed) and the job IDs of any jobs associated with the specified advance reservation, sorted by status.

-c all | **"***policy_name ...***"**

Shows advance reservation policies defined in `lsb.resources`. By default, displays all policy names.

The `all` keyword shows detailed information for all policies.

-p all | **"***host_name ...***"**

Shows a weekly planner for specified hosts using advance reservations.

The `all` keyword shows a weekly planner for all hosts with reservations.

-w

Wide format. Displays reservation information without truncating fields.

-h

Prints command usage and exits.

-V

Prints LSF release version and exits.

EXAMPLE

```
% brsvs -c reservation1
Policy Name: reservation1
Users: ugroup1 ~user1
Hosts: hostA hostB
Time Window: 8:00-13:00
```

SEE ALSO

`brsvadd`(8), `brsvdel`(8), `lsb.resources`(5)

brun

forces a job to run immediately

SYNOPSIS

brun [**-b**] [**-c**] [**-f**] **-m** **"***host_name***[#***num_cpus***]** ... **"** *job_ID*

brun [**-b**] [**-c**] [**-f**] **-m** **"***host_name***[#***num_cpus***]** ... **"** **"job_ID[***index_list***]"**

brun [**-h** | **-V**]

DESCRIPTION

This command can only be used by LSF administrators.

Forces a pending job to run immediately on specified hosts.

A job which has been forced to run is counted as a running job, this may 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 which 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 `brun` 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 `-c` 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. MXJ in `lsb.hosts` must be less than or equal to the number of CPUs and PJOB_LIMIT=1 must be specified in the queue (`lsb.queues`).

For example, a 6-CPU job is submitted to `hostA` and `hostB` with 4 CPUs each. Without `-c`, LSF would let 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 will remain pending. With -c, 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

Allows the job to run without being suspended due to run windows or suspending conditions.

-m "*host_name*[#*num_cpus*] ... "

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. *num_cpus* distributes job slots according the number of CPUs on the host. If *num_cpus* is not defined, or if *num_cpus* 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, or the number of free CPUs on the host if -c is specified.

***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.

brun does not guarantee a job will run; it just forces LSF to dispatch the job.

In the MultiCluster job forwarding model, you can only force a job by running the command in the execution cluster.

bsla

displays information about service class configuration for goal-oriented service-level agreement (SLA) scheduling

SYNOPSIS

bsla [*service_class_name*]

bsla [**-h** | **-V**]

DESCRIPTION

`bsla` displays the properties of service classes configured in `LSB_CONFDIR/`*`cluster_name`*`/configdir/lsb.serviceclasses` (see `lsb.serviceclasses`(5)) and dynamic information about the state of each configured service class.

OPTIONS

`service_class_name`

The name of a service class configured in `lsb.serviceclasses`.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

A list of job groups 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:

- THROUGHPUT
- VELOCITY
- DEADLINE

ACTIVE WINDOW

The configured time window when the service class goal is active. If a throughput or velocity goal has no time window configured, ACTIVE WINDOW is Always Open.

STATUS

Current status of the service class goal:

- Active:On time—the goal is active and meeting its target.
- Active:Delayed—the goal is active but is missing its target.
- Inactive—the goal is not active; 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 will be before the configured deadline. If the service class status is delayed, the service class is missing its goal and `bsla` 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 should be running in the service class for the SLA to meet its goal.

NJOBS

The current number of job slots used by jobs in the specified service class. A parallel job is counted as 1 job, regardless of the number of job slots it will use.

PEND

The number of pending job slots used by jobs in the specified service class.

RUN

The number of job slots used by running jobs in the specified service class.

SSUSP

The number of job slots used by the system-suspended jobs in the service class.

USUSP

The number of job slots used by user-suspended jobs in the specified service class.

FINISH

The number of jobs in the specified service class in EXITED or DONE state.

EXAMPLE

For the following service class named `Kyuquot` is configured in `lsb.serviceclasses`:

```
Begin ServiceClass
NAME = Kyuquot
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
```

`bsla` shows the following properties and current status:

```
% bsla Kyuquot
SERVICE CLASS NAME:  Kyuquot
 -- Daytime/Nighttime SLA
PRIORITY:  23
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
```

SEE ALSO

`bacct`(1), `bhist`(1), `bjobs`(1), `bkill`(1), `bmod`(1), `bsub`(1), `lsb.acct`(5), `lsb.serviceclasses`(5)

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 a you specify a job ID:

- You can get the external job status of jobs submitted by other users, but you cannot set job status of jobs submitted by other users.
- You can only set external status on your own jobs.
- Only root and LSF administrators can set external job status on jobs submitted by other users.

Job names are not unique; if you specify `-J` *job_name*:

- You can only get or set the external status on your own jobs.
- You cannot get or set external job status on jobs submitted by other users.
- Root and the LSF administrators can only get or set the external status 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.

-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`(1), `bread`(1)

bstop

suspends unfinished jobs

SYNOPSIS

bstop [**-a**] [**-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**] [**0**] [*job_ID* ... | **"***job_ID***[***index***]"**] ...

bstop [**-h** | **-V**]

DESCRIPTION

Suspends unfinished jobs.

Sends the SIGSTOP signal to sequential jobs and the SIGTSTP signal to parallel jobs to suspend them.

You must specify a job ID or `-g`, `-J`, `-m`, `-u`, or `-q`. 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 `bresume` to resume suspended jobs.

Using `bstop` on a job that is in the USUSP state has no effect.

You can also use `bkill -s STOP` to send the suspend signal to a job or use `bkill -s TSTP` to suspend one or more parallel jobs. Use `bkill -s CONT` to send a resume signal to a job.

If a signal request fails to reach the job execution host, LSF will retry 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.

-g *job_group_name*

Suspends only on jobs in the job group specified by *job_group_name*.

You cannot use `-g` with `-sla`. A job can either be attached to a job group or a service class, but not both.

-J *job_name*

Suspends only jobs with the specified name.

-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.

You cannot use -g with -sla. A job can either be attached to a job group or a service class, but not both.

Use bsla to display the properties of service classes configured in LSB_CONFDIR/*cluster_name*/configdir/lsb.serviceclasses (see lsb.serviceclasses(5)) 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.

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 invoker'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.

SEE ALSO

bsub(1), bjobs(1), bqueues(1), bhosts(1), bresume(1), bkill(1), bgadd(1), bgdel(1), bjgroup(1), bparams(5), mbatchd(8), kill(1), signal(2) lsb.params(5)

bsub

submits a batch job to LSF

SYNOPSIS

bsub [*options*] *command* [*arguments*]

bsub [**-h** | **-V**]

OPTION LIST

-B
-H
-I | **-Ip** | **-Is**
-K
-N
-r
-x
-a *esub_parameters*
-b [[*month***:**]*day***:**]*hour***:***minute*
-c [*hour***:**]*minute*[**/***host_name* | **/***host_model*] 139
-C *core_limit*
-D *data_limit*
-e *err_file*
-eo *err_file*
-ext[**sched**] **"***external_scheduler_options***"**
-E **"***pre_exec_command* [*arguments* ...]**"**
-f **"***local_file operator* [*remote_file*]**"** ...
-F *file_limit*
-g *job_group_name*
-G *user_group*
-i *input_file* | **-is** *input_file*
-J *job_name* | **-J** **"***job_name***[***index_list***]%***job_slot_limit***"**
-k **"***checkpoint_dir* [*checkpoint_period*][**method=***method_name*]**"**
-L *login_shell*
-Lp *ls_project_name*
-m **"***host_name*[**@***cluster_name*][**+**[*pref_level*]] | *host_group*[**+**[*pref_level*]] ...**"**
-M *mem_limit*
-n *min_proc*[**,***max_proc*]
-o *out_file*
-oo *out_file*
-P *project_name*
-p *process_limit*
-q **"***queue_name* ...**"**
-R **"***res_req***"**
-s *signal*
-S *stack_limit*
-sla *service_class_name*
-sp *priority*
-t [[*month***:**]*day***:**]*hour***:***minute*

-T *thread_limit*
-U *reservation_ID*
-u *mail_user*
-v *swap_limit*
-w **'***dependency_expression***'**
-W [*hour***:**]*minute*[**/***host_name* | **/***host_model*]
-wa **'**[*signal* | *command* | **CHKPNT**]**'**
-wt **'**[*hour***:**]*minute***'**
-Zs
-h
-V

DESCRIPTION

Submits a job for batch 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, 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 starting the job.

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. The job is run in `/tmp` if the current working directory is not accessible on the execution host.

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.

Use `-g` to submit a job to a job group.

Use `-n` to submit a parallel job.

Use `-I`, `-Is`, or `-Ip` to submit a batch interactive job.

Use `-J` to assign a name to your job.

Use `-k` to specify a checkpointable job.

To kill a batch job submitted with `bsub`, use `bkill`.

Jobs 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 Platform LSF on Windows* for more details.

Use `bmod` to modify jobs submitted with `bsub`. `bmod` takes similar options to `bsub`.

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, a job submitted by a given user will run under the same user's account on the execution host. For situations where nonuniform user names and user ID spaces exist, account mapping must be used to determine the account used to run a job.

`bsub` uses the command name as the job name. Quotation marks are significant.

If fairshare is defined and you belong to multiple user groups, the job will be 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, the queue is selected from your list. If LSB_DEFAULTQUEUE is not defined, the queue is selected from the system default queue list specified by the LSF administrator (see the parameter DEFAULT_QUEUE in `lsb.params(5)`).

LSF tries to obtain resource requirement information for the job from the remote task list that is maintained by the load sharing library (see `lsfintro(1)`). 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 (see `lshosts(1)`) as the submission host.

`bsub` assumes 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 batch 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 you define by setting the environment variable LSB_DEFAULTPROJECT. If you do not set LSB_DEFAULTPROJECT, the default project is the project specified by the LSF administrator in the `lsb.params` configuration file (see the

DEFAULT_PROJECT parameter in `lsb.params(5)`). If DEFAULT_PROJECT is not defined, then LSF uses `default` as the default project name.

OPTIONS

-B

Sends mail to you when the job is dispatched and begins execution.

-H

Holds the job in the PSUSP state when the job is submitted. The job will not be scheduled until you tell the system to resume the job (see `bresume(1)`).

-I | -Ip | -Is

-I

Submits a batch 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.

Terminal support is available for a batch interactive job.

When you specify the `-Ip` option, submits a batch interactive job and creates a pseudo-terminal when the job starts. Some applications (for example, `vi`) require a pseudo-terminal in order to run correctly.

When you specify the `-Is` option, submits a batch interactive job and creates a pseudo-terminal with shell mode support when the job starts. This option should be specified for submitting interactive shells, or applications which redefine the CTRL-C and CTRL-Z keys (for example, `jove`).

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.

You cannot use `-I`, `-Ip`, or `-Is` with the `-K` option.

Interactive jobs cannot be checkpointed.

Interactive jobs cannot be rerunnable (`bsub -r`).

The options that create a pseudo-terminal (`-Ip` and `-Is`) are not supported on Windows.

-K

Submits a batch job and waits for the job to complete. 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.

You will not be able to submit another job until the job is completed. This is useful when completion of the job is required in order 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` will exit 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.

You cannot use the `-K` option with the `-I`, `-Ip`, or `-Is` options.

-N

Sends the job report to you by mail when the job finishes. 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.

-r

If the execution host becomes unavailable while a job is running, specifies that the job will 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 will be requeued when the system restarts.

Reruns a job if the execution host or the system fails; it does not rerun a job if the job itself fails.

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`) cannot be rerunnable.

-x

Puts the host running your job into exclusive execution mode.

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(1)` reports the host status as closed_Excl, and `lsload(1)` 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(1)`, `lsrun(1)` or `lsgrun(1)` or any other load sharing applications.

You can force other batch 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(1)` or `lsgrun(1)` to explicitly specify the locked host.

-a *esub_parameters*

String format parameter containing the name of an application-specific `esub` program to be passed to the master `esub`. The master `esub` program (LSF_SERVERDIR/`mesub`) handles job submission requirements of the applications. Application-specific `esub` programs can specify their own job submission requirements. The value of `-a` is set in the LSB_SUB_ADDITIONAL option in the LSB_SUB_PARM file used by `esub`.

Use the `-a` option to specify which application-specific `esub` is invoked by `mesub`.

For example, to submit a job to `hostA` that invokes two application-specific `esub` programs named `esub.license`: and `esub.fluent`:

% **bsub -a license fluent -m hostA my_job**

`mesub` uses the method name `license` to invoke the `esub` named `LSF_SERVERDIR/esub.license`, and the method name `fluent` to invoke the `esub` named `LSF_SERVERDIR/esub.fluent`.

The value of `-a` is passed to `esub`, but it does not directly affect the other `bsub` parameters or behavior. The value of `-a` must correspond to an actual `esub` file. For example, to use `bsub -a fluent`, the file `esub.fluent` must exist in LSF_SERVERDIR.

Mandatory `esub` methods specified by LSB_ESUB_METHOD (environment variable or set in `lsf.conf`), are invoked before any esub programs specified by `-a`.

The name of the `esub` program must be a valid file name. It can contain only alphanumeric characters, underscore (_) and hyphen (-).

Compatibility note

After LSF version 5.1, the value of -a and LSB_ESUB_METHOD must correspond to an actual esub file in LSF_SERVERDIR. For example, to use bsub -a fluent, the file esub.fluent must exist in LSF_SERVERDIR.

-b [[*month*:]*day*:]*hour*:*minute*

Dispatches the job for execution on or after the specified date and time. The date and time are in the form of [[*month*:]*day*:]*hour*:*minute* where the number ranges are as follows: 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*, and four fields are assumed to be *month*:*day*:*hour*:*minute*.

-c [*hour*:]*minute*[/*host_name* | /*host_model*]

Limits the total CPU time the job can use. 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. (See `lsinfo`(1) to get host model information.) 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.

-C ***core_limit***

Sets a per-process (soft) core file size limit for all the processes that belong to this batch job (see `getrlimit(2)`). The core limit is specified in KB.

The behavior of this option depends on platform-specific UNIX 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.

-D ***data_limit***

Sets a per-process (soft) data segment size limit for each of the processes that belong to the batch job (see `getrlimit`(2)). The data limit is specified in KB. A `sbrk` call to extend the data segment beyond the data limit will return an error.

-e ***err_file***

Specify a file path. Appends the standard error output of the job to the specified file.

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/`.

-eo *err_file*

Specify a file path. Overwrites the standard error output of the job to the specified file.

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 will occur 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/`.

-ext[**sched**] **"***external_scheduler_options***"**

Application-specific external scheduling options for the job.

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, SGI IRIX hosts and AlphaServer SC hosts running RMS can exist in the same cluster, but they accept different external scheduler options. Use external scheduler options to define job requirements for either IRIX cpusets OR RMS, but *not* both. Your job will run either on IRIX or RMS. If external scheduler options are not defined, the job may run on IRIX but it will not run on an RMS 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.

See *Using Platform LSF HPC* for information about specific external scheduler options.

-E "*pre_exec_command [arguments ...]***"**

Runs the specified pre-exec command on the batch job's execution host before actually running the job. For a parallel job, the pre-exec command runs on the first host selected for the parallel job.

If the pre-exec command exits with 0 (zero), then the real job is started on the selected host. Otherwise, the job (including the pre-exec command) goes back to PEND status and is rescheduled.

If your job goes back into PEND status, LSF will keep on trying to run the pre-exec command and the real job when conditions permit. For this reason, be sure that your pre-exec command can be run many times without having side effects.

The standard input and output for the pre-exec command are directed to the same files as for the real job. The pre-exec command runs under the same user ID, environment, home, and working directory as the real job. If the pre-exec command is not in the user's normal execution path (the $PATH variable), the full path name of the command must be specified.

-f "*local_file operator* [*remote_file*]" ...

Copies a file between the local (submission) host and the remote (execution) host. 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.

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.

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 will be 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.

-F *file_limit*

Sets a per-process (soft) file size limit for each of the processes that belong to the batch job (see `getrlimit`(2)). The file size 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.

-g *job_group_name*

Submits jobs in the job group specified by *job_group_name* 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.

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 cannot use `-g` with `-sla`. A job can either be attached to a job group or a service class, but not both.

Job group names can be up to 512 characters long.

-G *user_group*

Only useful with fairshare scheduling.

Associates the job with the specified group. Specify any group that you belong to that does not contain any subgroups. You must be a direct member of the specified user group.

-i *input_file* | **-is** *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 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, 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.

If JOB_SPOOL_DIR in `lsb.params` 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 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 -is` cannot write to the default directory `LSB_SHAREDIR/`*`cluster_name`*`/lsf_indir` and the job will fail.

Unless you use `-is`, 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). The special characters `%J` and `%I` are not valid with the `-is` option.

-J *job_name* | **-J "***job_name***[***index_list***]%***job_slot_limit***"**

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.

The job name does not need to be unique.

The job name can be up to 512 characters long.

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. The job array index starts at one. By default, the maximum job array index is 1000.

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.

-k "*checkpoint_dir* [*checkpoint_period*][**method=***method_name*]**"**

Makes a job checkpointable and specifies the checkpoint directory. If you omit the checkpoint period, the quotes are not required. Specify a relative or absolute path 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)).

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`(1). Because checkpointing is a heavyweight operation, you should choose a checkpoint period greater than half an hour.

Optionally, specifies a custom checkpoint and restart method to use with the job. Use `method=default` to indicate to use the default LSF checkpoint and restart programs for the job, `echkpnt.default` and `erestart.default`.

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 *login_shell*

Initializes the execution environment using the specified login shell. The specified login shell must be an absolute path. This is not necessarily the shell under which the job will be executed.

Login shell is not supported on Windows.

-Lp *ls_project_name*

Assigns the job to the specified License Scheduler project.

-m "*host_name*[**@***cluster_name*][**+**[*pref_level*]] | *host_group*[**+**[*pref_level*]] **..."**

Runs the job on one of the specified hosts.

By default, if multiple hosts are candidates, runs the job on the least-loaded host.

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 all the hosts in the your host list. Otherwise, the job is not submitted. To find out what hosts are configured for the queue, use `bqueues -l`.

To display configured host groups, use `bmgroup`.

For the MultiCluster job forwarding model, you cannot specify a remote host by name.

-M *mem_limit*

Sets a per-process (soft) memory limit for all the processes that belong to this batch job (see `getrlimit(2)`). The memory limit is specified in KB.

If LSB_MEMLIMIT_ENFORCE or LSB_JOB_MEMLIMIT are set to `y` in `lsf.conf`, LSF kills the job when it exceeds the memory limit. Otherwise, LSF passes the memory limit to the operating system. UNIX operating systems that support RUSAGE_RSS for `setrlimit()` can apply the memory limit to each process.

The following operating systems do not support the memory limit at the OS level:

- Windows

- Sun Solaris 2.x

-n *min_proc*[,*max_proc*]

Submits a parallel job and specifies the number of processors required to run the job (some of the processors may be on the same multiprocessor host).

You can specify a minimum and maximum number of processors to use. The job can start if at least the minimum number of processors is available. If you do not specify a maximum, the number you specify represents the exact number of processors to use.

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.

Jobs that request fewer slots than the minimum PROCLIMIT defined for the queue to which the job is submitted, or more slots than the maximum PROCLIMIT cannot use the queue and are rejected. If the job requests minimum and maximum job slots, the maximum slots requested cannot be less than the minimum PROCLIMIT, and the minimum slots requested cannot be more than the maximum PROCLIMIT.

For example, if the queue defines PROCLIMIT=4 8:

- `bsub -n 6` is accepted because it requests slots within the range of PROCLIMIT
- `bsub -n 7` is rejected because it requests more slots than the PROCLIMIT allows
- `bsub -n 1` is rejected because it requests fewer slots than the PROCLIMIT allows
- `bsub -n 6,10` is accepted because the minimum value 6 is within the range of the PROCLIMIT setting
- `bsub -n 1,6` is accepted because the maximum value 6 is within the range of the PROCLIMIT setting
- `bsub -n 10,16` is rejected because its range is outside the range of PROCLIMIT
- `bsub -n 1,3` is rejected because its range is outside the range of PROCLIMIT

See the PROCLIMIT parameter in `lsb.queues`(5) for more information.

In a MultiCluster environment, if a queue exports jobs to remote clusters (see the SNDJOBS_TO parameter in `lsb.queues`(5)), then the process limit is not imposed on jobs submitted to this queue.

Once at 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.

-o *out_file*

Specify a file path. Appends the standard output of the job to the specified file. 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 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` 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 will advise you where to find your output.

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).

-oo *out_file*

Specify a file path. 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 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 will occur 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 will advise you where to find your output.

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).

-P *project_name*

Assigns the job to the specified project.

On IRIX 6, you must be a member of the project as listed in `/etc/project`(4). If you are a member of the project, then `/etc/projid`(4) maps the project name to a numeric project ID. Before the submitted job executes, a new array session (`newarraysess`(2)) is created and the project ID is assigned to it using `setprid`(2).

-p *process_limit*

Sets the limit of the number of processes to *process_limit* for the whole job. The default is no limit. Exceeding the limit causes the job to terminate.

-q "*queue_name ...***"**

Submits the job to one of the specified queues. 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 selects the most appropriate queue in the list for your job based on the job's resource limits, and other restrictions, such as the requested hosts, your accessibility to a queue, queue status (closed or open), etc. The order in which the queues are considered is the same order in which these queues are listed. The queue listed first is considered first.

-R "*res_req***"**

Runs the job on a host that meets the specified resource requirements. A resource requirement string describes the resources a job needs. LSF uses resource requirements to select hosts for remote execution and job execution.

The size of the resource requirement string is limited to 512 characters.

Any run-queue-length-specific resource, such as `r15s`, `r1m` or `r15m`, specified in the resource requirements refers to the normalized run queue length.

A resource requirement string is divided into the following sections:

- A selection section (`select`). The selection section specifies the criteria for selecting 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 batch 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.

If no section name is given, then the entire string is treated as a selection string. The `select` keyword may be omitted if the selection string is the first string in the resource requirement.

The resource requirement string has the following syntax:

```
select[selection_string] order[order_string]
rusage[usage_string [, usage_string][|| usage_string] ...]
span[span_string] same[same_string]
```

The square brackets must be typed as shown.

The section names are `select`, `order`, `rusage`, `span`, and `same`. Sections that do not apply for a command are ignored.

Each section has a different syntax.

For example, to submit a job which will run on Solaris 7 or Solaris 8:

```
% bsub -R "sol7 || sol8" 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[cpu]" myjob
```

You defined a resource 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 job that will run on `hostE`:

```
% bsub -R "bigmem" myjob
```

or

```
% bsub -R "defined(bigmem)" myjob
```

You configured a static shared resource for licenses for the Verilog application as a resource called `verilog_lic`. To submit a job that will run 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 and 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
```

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 will not work with 2.0.1.

Job-level resource requirement specifications that use the `||` 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 will only use `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 will try `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 will use 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`**

-s *signal*

Send the specified signal when a queue-level run window closes.

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.

-S *stack_limit*

Sets a per-process (soft) stack segment size limit for each of the processes that belong to the batch job (see `getrlimit`(2)). The limit is specified in KB.

-sla *service_class_name*

Specifies the service class where the job is to run.

If the SLA does not exist or the user is not a member of the service class, the job is rejected.

You cannot use `-sla` with `-g`. A job can either be attached to a job group or a service class, but not both.

You should submit your jobs with a run time limit (`-W` option) or the queue should specify a run time limit (RUNLIMIT in the queue definition in `lsb.queues`). If you do not specify a run time limit, 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`(5)) and dynamic information about the state of each service class.

-sp *priority*

Specifies user-assigned job priority which allow users to order their jobs in a queue. Valid values for priority are any integers between 1 and MAX_USER_PRIORITY (displayed by `bparams -l`). Job priorities that are not valid are rejected. LSF and queue administrators can specify priorities beyond MAX_USER_PRIORITY.

The job owner can change the priority of their own jobs. LSF 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 with the same priority are ordered first come first served.

User-assigned job priority can be configured with automatic job priority escalation to automatically increase the priority of jobs that have been pending for a specified period of time.

-t [[*month*:]*day*:]*hour*:*minute*

Specifies the job termination deadline.

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)).

The format for the termination time is [[*month*:]*day*:]*hour*:*minute* where the number ranges are as follows: 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*, and four fields are assumed to be *month*:*day*:*hour*:*minute*.

-T *thread_limit*

Sets the limit of the number of concurrent threads to *thread_limit* for the whole job. 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.

-U *reservation_ID*

If an advance reservation has been created with the `brsvadd` command, the `-U` option makes use of the reservation.

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, we assume the default queue is configured to forward jobs to the remote cluster.

-u *mail_user*

Sends mail to the specified email destination.

-v *swap_limit*

Set the total process virtual memory limit to *swap_limit* in KB for the whole job. The default is no limit. Exceeding the limit causes the job to terminate.

-w **'***dependency_expression***'**

LSF will not place your job unless the dependency expression evaluates to TRUE. 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 dependency conditions, job names specify only your own jobs, unless you are the LSF administrator. 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 will be 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.

done(*job_ID* |**"***job_name***"** ...**)**

The job state is DONE.

LSF refers to the oldest job of *job_name* in memory.

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.

If you specify the first word of the message 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 (the post-processing of specified job has completed without errors).

post_err(*job_ID* | **"***job_name***")**

The job state is POST_ERR (the post-processing of the specified job has completed with errors).

started(*job_ID* | **"***job_name***")**

The job state is:

- RUN, DONE, or EXIT

- PEND or PSUSP, and the job has a pre-execution command (**bsub -E**) that is running.

-W [*hour*:]*minute*[/*host_name* | /*host_model*]

Sets the run time limit of the batch job. 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`(5)).

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 run time limit is not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted with a run limit.

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. (See `lsinfo`(1) to get host model information.)

If no host or host model is given, LSF uses the default run time 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 MultiCluster 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 will be aborted.

If the IGNORE_DEADLINE parameter is set in `lsb.queues`(5), 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.

-wa '[*signal* | *command* | **CHKPNT**]'

Specifies the job action to be taken before a job control action occurs.

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.

You can specify actions similar to the JOB_CONTROLS queue level parameter: send a signal, invoke a command, or checkpoint the job.

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.

For example the following specifies that 2 minutes before the job reaches its run time limit, an URG signal is sent to the job:

```
% bsub -W 60 -wt '2' -wa 'URG' myjob
```

-wt '[*hour*:]*minute*'

Specifies the amount of time before a job control action occurs that a job warning action is to be taken. 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.

For example the following specifies that 2 minutes before the job reaches its run time limit, an URG signal is sent to the job:

```
% bsub -W 60 -wt '2' -wa 'URG' myjob
```

-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.

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 in `lsb.params` is specified, the `-is` option spools the command file to the specified directory and uses the spooled file as the input file for the job.

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 -is` cannot write to the default directory `LSB_SHAREDIR/`*`cluster_name`*`/lsf_cmddir` and the job will fail.

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.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

`command [argument]`

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*.

If the batch 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`; e.g., `#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`.

See EXAMPLES below for examples of specifying commands through standard input.

OUTPUT

If the job is successfully submitted, displays the job ID and the queue to which the job has been submitted.

EXAMPLES

% **bsub sleep 100**

Submit the UNIX command `sleep` together with its argument `100` as a batch job.

% **bsub -q short -o my_output_file "pwd; ls"**

Submit the UNIX command `pwd` and `ls` as a batch job to the queue named `short` and store the job output in `my_output` file.

% **bsub -m "host1 host3 host8 host9" my_program**

Submit `my_program` to run on one of the candidate hosts: host1, host3, host8 and host9.

% **bsub -q "queue1 queue2 queue3" -c 5 my_program**

Submit `my_program` to one of the candidate queues: `queue1`, `queue2`, and `queue3` which are selected according to the CPU time limit specified by `-c 5`.

% **bsub -I ls**

Submit a batch interactive job which displays the output of `ls` at the user's terminal.

% **bsub -Ip vi myfile**

Submit a batch interactive job to edit `myfile`.

% **bsub -Is csh**

Submit a batch interactive job that starts `csh` as an interactive shell.

% 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.

% bsub my_script

Submit `my_script` as a batch 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 will be 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 will be run under `/bin/csh`.

% bsub -q night < my_script

where `my_script` contains:

```
#!/bin/sh
#BSUB -q test
#BSUB -o outfile -e errfile # my default stdout, stderr
files
#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 -b 20:00 -J my_job_name

```
bsub> sleep 1800
bsub> my_program
bsub> CTRL-D
```

The job commands are entered interactively.

% bsub -T 4 myjob

Submits `myjob` with a maximum number of concurrent threads of 4.

% bsub -W 15 -sla Kyuquot sleep 100

Submit the UNIX command `sleep` together with its argument `100` as a batch job to the service class named `Kyuquot`.

LIMITATIONS

When using account mapping the command `bpeek`(1) will not work. File transfer via the `-f` option to `bsub`(1) requires `rcp`(1) to be working between the submission and execution hosts. Use the `-N` option to request mail, and/or the `-o` and `-e` options to specify an output file and error file, respectively.

SEE ALSO

`bjobs`(1), `bkill`(1), `bqueues`(1), `bhosts`(1), `bmgroup`(1), `bmod`(1), `bchkpnt`(1), `brestart`(1), `bgadd`(1), `bgdel`(1), `bjgroup`(1), `sh`(1), `getrlimit`(2), `sbrk`(2), `libckpt.a`(3), `lsb.users`(5), `lsb.queues`(5), `lsb.params`(5), `lsb.hosts`(5), `lsb.serviceclasses`(5), `mbatchd`(8)

bswitch

switches unfinished jobs from one queue to another

SYNOPSIS

bswitch [**-J** *job_name*] [**-m** *host_name* | **-m** *host_group*] [**-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 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 has been 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.

If a switched job has not been dispatched, then its behavior will be as if it were submitted to the new queue in the first place.

If a switched job has been dispatched, then it will be controlled by the `loadSched` and `loadStop` vectors and other configuration parameters of the new queue, but its nice value and resource limits will remain the same.

Also, if a switched job has been dispatched, it will be 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 WAIT jobs are requeued to PEND. 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 inherited from the queue.

OPTIONS

0

(Zero). Switches multiple jobs. Switches all the jobs that satisfy other specified options (`-m`, `-q`, `-u` and `-J`).

-J *job_name*

Only switches jobs that have the specified job name.

-m *host_name* | **-m** *host_group*

Only switches jobs dispatched to the specified host or host group.

-q *queue_name*

Only switches jobs in the specified queue.

-u *user_name* | **-u** *user_group* | **-u all**

Only switches jobs submitted by the specified user, or all users if you specify the keyword `all`.

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 MultiCluster job.

SEE ALSO

`bqueues`(1), `bhosts`(1), `bugroup`(1), `bsub`(1), `bjobs`(1)

btop

moves a pending job relative to the first job in the queue

SYNOPSIS

btop *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.

The `btop` command allows users and the LSF administrator to manually change the order in which jobs are considered for dispatch. Users can only operate on their own jobs, whereas the LSF administrator can operate on any user's jobs. Users can only change the relative position of their own jobs.

If invoked by the LSF administrator, `btop` moves the selected job before the first job with the same priority submitted to the queue. The positions of all users' jobs in the queue can be changed by the LSF administrator.

If invoked by a regular user, `btop` moves the selected job before the first job with the same priority submitted by the user to the queue. Pending jobs are displayed by `bjobs` in the order in which they will be considered for dispatch.

A user may use `btop` to change the dispatch order of his/her jobs scheduled using a fairshare policy. However, if a job scheduled using a fairshare policy is moved by the LSF administrator using `btop`, the job will not be subject to further fairshare scheduling unless the same job is subsequently moved by the LSF administrator using `bbot`; in this case the job will be scheduled again using the same fairshare policy (see the `FAIRSHARE` keyword in `lsb.queues`(5) and `HostPartition` keyword in `lsb.hosts` (5)).

To prevent users from changing the queue position of a pending job with `btop`, configure `JOB_POSITION_CONTROL_BY_ADMIN=Y` in `lsb.params`.

OPTIONS

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 share 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. *position* 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`(1), `bjobs`(1), `bswitch`(1)

bugroup

displays information about user groups

SYNOPSIS

bugroup [**-l**] [**-r**] [**-w**] [*user_group* ...]

bugroup [**-h** | **-V**]

DESCRIPTION

Displays user groups and user names for each group.

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 ...

Only displays information about the specified user groups. Do not use quotes when specifying 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 and user shares are defined in the configuration file `lsb.users`(5).

SEE ALSO

`lsb.users`(5), `bmgroup`(1), `busers`(1)

busers

displays information about users and user groups

SYNOPSIS

busers [**-w**] [*user_name* ... | *user_group* ... | **all**]

busers [**-h** | **-V**]

DESCRIPTION

Displays information about users and user groups.

By default, displays information about the user who runs the command.

OPTIONS

user_name ... | *user_group ...* | **all**

Displays information about the specified users or user groups, or about all users if you specify `all`.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

-w

Prints user and user group pending job thresholds 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 which have jobs slots reserved for them. For preemptive scheduling, these job slots are used by running jobs or by pending jobs which have slots reserved for them. (see the description of PREEMPTION in `lsb.queues`(5)). This job limit is configured per processor so that multiprocessor hosts have more job slots. If the dash character (-) is displayed, there is no limit. JL/P is defined in the LSF configuration file lsb.users(5).

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 which have job slots reserved for them. For preemptive scheduling, these job slots are used by running jobs or by pending jobs which have job slots reserved for them. (see the description of

PREEMPTIVE in `lsb.queues`(5)). If the character '–' is displayed, there is no limit. MAX is defined by the MAX_JOBS parameter in the configuration file `lsb.users(5)`.

NJOBS

The current number of job slots used by specified users' jobs. A parallel job that is pending is counted as *n* job slots for it will use *n* job slots in the queue when it is dispatched.

PEND

The number of pending job slots used by jobs of the specified users.

RUN

The number of job slots used by running jobs of the specified users.

SSUSP

The number of job slots used by the system-suspended jobs of the specified users.

USUSP

The number of job slots used by user-suspended jobs of the specified users.

RSV

The number of job slots used by pending jobs of the specified users which have job slots reserved for them.

MPEND

The pending job threshold for the specified users or user groups. MPEND is defined by the MAX_PEND_JOBS parameter in the configuration file `lsb.users(5)`.

SEE ALSO

`bugroup`(1), `lsb.users`(5), `lsb.queues`(5)

ch

changes the host on which subsequent commands are to be executed

SYNOPSIS

ch [**-S**] [**-t**] [*host_name*]

ch [**-h** | **-V**]

DESCRIPTION

Changes the host on which subsequent commands are to be executed.

By default, if no arguments are specified, changes the current host to the home host, the host from which the `ch` command was issued.

By default, executes commands on the home host.

By default, shell mode support is not enabled.

By default, does not display execution time of tasks.

The `ch` command allows you 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 that of the parent shell. Every remotely dispatched command is executed with the same environment as that on the home host. The syntax of the `ch` command is similar to that of the Bourne shell. However, there are some important differences.

The ampersand (`&)` following a command line (representing a background job in the Bourne shell) is ignored by `ch`. You can submit background jobs in `ch` with the built-in `post` command and bring them into the foreground with the built-in `contact` command (see below for details).

`ch` recognizes a ~ (tilde) as a special path name. If a ~ (tilde) is followed by a space, tab, new line or `/` (slash) character, then the ~ character is translated into the user's home directory. Otherwise, the ~ is translated as the home directory of the user name given by the string following the ~ character. Pipelines, lists of commands and redirection of standard input/output are all handled by invoking `/bin/sh`.

The following sequence of commands illustrates the behavior of the `ch` command. For example, the user is currently 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 which 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 execute is displayed.

host_name

Executes 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. `ch` assigns a unique task ID to this command and displays this ID, then continues to interact with the user. However, the output of background jobs may disturb the screen. You can post multiple commands on one host or on different hosts. When a previously posted command is completed, `ch` reports its status to the standard error. If a command is not specified, `ch` displays all currently running background commands.

contact *task_ID*

Brings a previously posted background command into the foreground. *task_ID* is the ID returned by the `post` command. Standard input is now passed to this foreground command. You cannot put an active foreground job into the background. A command that has been brought into the foreground with the `contact` command cannot be put back into the background.

exit

Exits `ch` if there are no posted commands running. Typing an EOF character (usually CTRL-D but may be set otherwise, see `stty(1)`) forces `ch` to exit; uncompleted posted commands are killed.

LIMITATIONS

Currently, the `ch` command does not support script, history, nor alias.

The `ch` prompt is always the *current working host:current working directory* followed by a > (right angle bracket) character. If the `ch` session is invoked by a shell that supports job control (such as `tcsh` or `ksh`), `CTRL-Z` suspends the whole `ch` session. The exit status of a command line is printed to `stderr` if the status is non-zero.

SEE ALSO

`lsrun`(1), `rsh`(1), `stty`(1)

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 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 owned by the user who invoked the `lsacct` command.

By default, displays accounting statistics for tasks executed on all hosts in the LSF system.

By default, displays statistics for tasks logged in the task log file currently used by RES: `LSF_RES_ACCTDIR/lsf.acct.`*host_name* `or /tmp/lsf.acct.`*host_name* (see `lsf.acct`(5)).

If `-l` is not specified, the default is to display the fields in SUMMARY only (see OUTPUT).

The RES on each host writes its own accounting log file. These files can be merged using the `lsacctmrg` command to generate statistics for the entire LSF cluster.

All times are reported in seconds. All sizes are reported in kilobytes.

OPTIONS

-l

Per-task statistics. Displays statistics about each task. See OUTPUT for a description of information that is displayed.

-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 `bhist`(1).

-S *time0,time1*

Displays accounting statistics for only tasks that began executing during the specified time interval.

The time format is the same as in `bhist`(1).

-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 merged with the `lsacctmrg` command.

-m *host_name ...*

Displays accounting statistics for only tasks 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 (').

-u *user_name ...* | **-u all**

Displays accounting statistics for only tasks 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.

pid ...

Displays accounting statistics for only tasks with the specified *pid*. This option overrides all other options except for `-l`, `-f`, `-h`, `-V`.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

SUMMARY (default format)

Overall statistics for tasks.

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 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` (2).

CPU time

Total CPU time 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 (-l)

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, 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 executed.

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 `lsf.acct.`*host_name*

SEE ALSO

`lsf.acct`(5), `lsacctmrg`(1), `res`(8), `bhist`(1)

lsacctmrg

merges 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 specified in `lsf.acct` (see `lsf.acct`(5)).

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`(5), `res`(8)

lsadmin

administrative tool for LSF

SYNOPSIS

lsadmin *subcommand*

lsadmin [**-h** | **-V**]

SUBCOMMAND LIST

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 [*host_name* ... | **all**] [**-c** *cpu_time*]
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* ...**"**]
help [*subcommand* ...] | **?** [*subcommand* ...]
quit
-h
-V

DESCRIPTION

This command can only be used by LSF administrators.

`lsadmin` is a tool that executes privileged commands to control LIM and RES operations in the LSF cluster.

If no subcommands are supplied for `lsadmin`, `lsadmin` prompts for subcommands from the standard input.

For subcommands for which multiple host names or host groups can be specified, do not enclose the multiple names in quotation marks.

Obsolete commands

- The command `lslockhost`(8) is superseded by `lsadmin limlock`
- The command `lsreconfig`(8) is superseded by `lsadmin reconfig`
- The command `lsunlockhost`(8) is superseded by `lsadmin limunlock`

OPTIONS

subcommand

Executes the specified subcommand. See 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 LIMs on all hosts in the cluster. You should use `reconfig` after changing configuration files. The configuration files are checked before all LIMs in the cluster are restarted. If the configuration files are not correct, reconfiguration will not be initiated.

If LSF_MASTER_LIST is specified in `lsf.conf`, 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 fatal 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 promted to confirm LIM startup.

Only `root` and users listed in the parameter LSF_STARTUP_USERS in `lsf.sudoers`(5) can use the `all` and `-f` options to start LIM as `root`.

These users must also be able to use `rsh` or `ssh` on all LSF hosts without having to type in passwords. If permission to start up LIMs as `root` is not configured, `limstartup` will start up LIMs as yourself after your confirmation.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

-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 promted 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.

`limrestart` should be used with care. Do not make any modifications until all the LIMs have completed the startup process. If you execute `limrestart` *host_name*`...` to restart some of the LIMs after changing the configuration files, but other LIMs are still running the old configuration, confusion will arise among these LIMs. To avoid this situation, use `reconfig` instead of `limrestart`.

-v

Displays detailed messages about configuration file checking.

-f

Disables user interaction and forces LIM to restart if no fatal errors are found. This option is useful in batch mode. `limrestart -f all` is the same as `reconfig -f`.

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 will be sent to a locked host by LSF.

-l *time_seconds*

The host is locked for the specified time in seconds.

This is useful if a machine is running an exclusive job requiring all the available CPU time and/or memory.

limunlock

Unlocks LIM on the local host.

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

Starts RES on the local host if no arguments are specified.

Starts RESs 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 defined by the LSF_STARTUP_USERS parameter in `lsf.sudoers`(5) can use the `all` and `-f` options to start RES as `root`.

These users must be able to use `rsh` or `ssh` on all LSF hosts without having to type in passwords. For `root` installation to work properly, `lsadmin` must be installed as a `setuid` to `root` program.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

-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 RESs 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 will keep running until all remote tasks exit.

-f

Disables interaction and does not ask for confirmation for shutting down RESs.

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

Restarts RES on the local host if no arguments are specified.

Restarts RESs 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 will keep running until all remote tasks exit. While waiting 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 RESs.

reslogon [*host_name* ... | **all**] [**-c** *cpu_time*]

Logs all tasks executed by RES on the local host if no arguments are specified.

Logs tasks executed by RESs on the specified hosts or on all hosts in the cluster if `all` is specified.

RES will write the task's resource usage information into the log file `lsf.acct.`*host_name*. The location of the log file is determined by LSF_RES_ACCTDIR defined in `lsf.conf`. If LSF_RES_ACCTDIR is not defined, or RES cannot access it, the log file will be created in `/tmp` 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_name=current 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 command was submitted)

In MultiCluster, debug levels can only be set for hosts within the same cluster. For example, you could not 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. If a list of classes is specified, they must be enclosed in quotation marks and separated by spaces.

Possible classes:

LC_AFS - Log AFS messages

LC_AUTH - Log authentication messages

LC_CHKPNT - log checkpointing messages

LC_COMM - Log communication messages

LC_DCE - Log messages pertaining to DCE support

LC_EXEC - Log significant steps for job execution

LC_FILE - Log file transfer messages

LC_HANG - Mark where a program might hang

LC_LICENCE - Log license management messages

LC_MULTI - Log messages pertaining to MultiCluster

LC_PIM - Log PIM messages

LC_SIGNAL - Log messages pertaining to signals

LC_TRACE - Log significant program walk steps

LC_XDR - Log everything transferred by XDR

Default: 0 (no additional classes are logged)

Note: Classes are also listed in `lsf.h`.

-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.

Possible values:

`0` - LOG_DEBUG level in parameter LSF_LOG_MASK in `lsf.conf`.

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

2 - LOG_DEBUG2 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 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.

Default: 0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

-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 may be specified.

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

The name of the file created 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, no log file is created.

Default: current LSF system log file in the LSF system log file directory, in the format *daemon_name*.log.*host_name*.

-o

Turns off temporary debug settings and reset them to the daemon starting state. The message log level is reset back to the value of LSF_LOG_MASK and classes are reset to the value of LSF_DEBUG_RES, LSF_DEBUG_LIM.

Log file is reset back to the default log file.

"*host_name* ...**"**

Sets debug settings on the specified host or hosts.

Default: local host (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 LIM to include additional 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 host (host from which command was submitted)

In MultiCluster, timing levels can only be set for hosts within the same cluster. For example, you could not 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.

Valid values: 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.

Default: undefined (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 may be specified.

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

The name of the file created 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, no log file is created.

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

Default: current LSF system log file in the LSF system log file directory, in the format *daemon_name*.`log`.*host_name*.

-o

Turns off temporary timing settings and resets them to the daemon starting state. The timing level is reset back to the value of the parameter for the corresponding daemon (LSF_TIME_LIM, LSF_TIME_RES).

Log file is reset back to the default log file.

"*host_name* ...**"**

Sets the timing level on the specified host or hosts.

Default: 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 additional timing information in log files. You must be the LSF administrator can use this command, not root.

See description of `limtime` for an explanation of options.

help [*subcommand* ...] | **?** [*subcommand* ...]

Displays the syntax and functionality of the specified commands. The commands must be explicit to `lsadmin`.

From the command prompt, you may use `help` or `?`.

quit

Exits the `lsadmin` session.

SEE ALSO

`ls_limcontrol`(3), `ls_rescontrol`(3), `ls_readconfenv`(3), `ls_gethostinfo`(3), `ls_connect`(3), `ls_initrex`(3), `lsf.conf`(5), `lsf.sudoers`(5), `lsf.acct`(5), `bmgroup`(1), `busers`(1)

lsclusters

displays configuration information about LSF clusters

SYNOPSIS

lsclusters [**-l**] [*cluster_name* ...]

lsclusters [**-h** | **-V**]

DESCRIPTION

Displays configuration information about LSF clusters.

By default, returns information about the local cluster and all other clusters of which the local cluster is aware (all clusters defined in the `RemoteClusters` section of `lsf.cluster.`*cluster_name* if that section exists, otherwise all clusters defined in `lsf.shared`).

OPTIONS

-l

Long format. Displays additional information.

cluster_name ...

Only displays information about the specified clusters.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

Default Output

The information includes: cluster name, cluster master host, primary cluster administrator's login name, total number of hosts in the cluster, and the number of server hosts in the cluster.

A listing of the clusters is displayed with the following fields:

CLUSTER_NAME

The name of the cluster.

STATUS

The current status of the cluster. Possible values are:

ok

The cluster is in normal load sharing state, and will 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 hosts in the cluster.

SERVERS

Number of LSF server hosts in the cluster.

Long Format (-l)

If this option is specified, the command will also list available resource names, host types, host models and cluster administrator's login names, and whether local cluster accepts or sends interactive jobs to this cluster.

SEE ALSO

`lsfintro`(1), `ls_info`(3), `ls_policy`(3), `ls_clusterinfo`(3) `lsf.cluster`(5)

lseligible

displays whether a task is eligible for remote execution

SYNOPSIS

lseligible [**-r**] [**-q**] [**-s**] *task*

lseligible [**-h** | **-V**]

DESCRIPTION

Displays whether the specified task is eligible for remote execution.

By default, only tasks in the remote task list are considered eligible for remote execution.

OPTIONS

-r

Remote mode. Considers eligible for remote execution any task not included in the local task list.

-q

Quiet mode. Displays only the resource requirement string defined for the task. The string ELIGIBLE or NON-ELIGIBLE is omitted.

-s

Silent mode. No output is produced. The `-q` and `-s` options are useful for shell scripts which operate by testing the exit status (see DIAGNOSTICS).

task

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 associated with the task are printed to stdout. Otherwise, the string NON-ELIGIBLE is printed to stdout.

If `lseligible` prints ELIGIBLE with no resource requirements, the task has the default requirements of CPU consumption and memory usage.

DIAGNOSTICS

`lseligible` has the following exit statuses:

0 Task is eligible for remote execution

1 Command is to be executed locally

-1 Syntax errors

-10 A failure is detected in the LSF system

SEE ALSO

ls_eligible(3), lsrtasks(1), lsf.task(5)

lsfinstall

runs `lsfinstall`, the Platform LSF installation and configuration script

SYNOPSIS

`lsfinstall -f install.config`

`lsfinstall -s -f slave.config`

`lsfinstall -h`

DESCRIPTION

`lsfinstall` runs the LSF installation scripts and configuration utilities to install a new Platform LSF cluster or upgrade LSF from a previous release.

To install a fully operational LSF cluster that all users can access, you should install as root.

You can run `lsfinstall` as a non-root user, with limitations, described in "If you install as a non-root user" on page 204.

Required install.config variables

- LSF_TOP="/*path*"
- LSF_ADMINS="*user_name* [*user_name*...]"
- LSF_CLUSTER_NAME="*cluster_name*"

See "install.config" on page 307 for an example `install.config` file.

Required slave.config variables

If you use `slave.config` for dynamic slave host installation, the following parameters are required:

- LSF_TOP="/*path*"
- LSF_TARDIR="/*path*"
- LSF_SERVER_HOSTS="*host_name* [*host_name* ...]"

See "slave.config" on page 627 for an example `slave.config` file.

Variables that require an absolute path

- LSF_LICENSE="/*path*/*license_file*"
- LSF_TOP="/*path*"
- LSF_TARDIR="/*path*"

What lsfinstall does

Before installing and configuring LSF, `lsfinstall` checks the installation prerequisites, and outputs the results to `lsfprechk.rpt`. `lsfinstall` writes any unrecoverable errors to the `Install.err file` and exits. You must correct these errors before continuing to install and configure LSF.

During installation, `lsfinstall` logs installation progress in the `Install.log` file, calls other utilities to uncompress, extract and copy product files, installs a license, and configures the cluster.

After installation, you should run `hostsetup` to set up each server host in the cluster. After setting up the server hosts, you should start your cluster and test the installation by running some basic commands.

Where lsfinstall is located

`lsfinstall` is included in the LSF installation script tar file `lsf6.2_lsfinstall.tar.Z` and is located in the `lsf6.2_lsfinstall` directory created when you uncompress and extract installation script tar file.

After installation, `lsfinstall` is located in `LSF_TOP/6.2/install/`.

Before running lsfinstall

1. Plan your installation by choosing:
   - LSF installation directory on file server (e.g., `LSF_TOP="/usr/share/lsf"`)
   - LSF hosts (master host, server hosts, and client-only hosts; e.g., `LSF_ADDSERVERS="hosta hostb hostc"`)
   - Cluster name (39 characters or less with no white spaces; e.g., `LSF_CLUSTER_NAME="cluster1"`)

     Do not use the name of any host, user, or user group as the name of your cluster.

   - Primary LSF administrator (owns the LSF configuration files and log files; e.g., `LSF_ADMINS="lsfadmin"`)
   - LSF server hosts that are candidates to become the master host for the cluster, if you are installing a new host to be dynamically added to the cluster (e.g., `LSF_MASTER_LIST="hosta hostb"`)
2. Prepare your systems for installation:
   - Make sure the installation file system on the file server host has enough disk space for all hosts types (approximately 300 MB per host type).
   - Make sure the top-level installation directory (LSF_TOP) is accessible with the same path name from all hosts in the cluster (e.g., `/usr/share/lsf`).
   - Create user accounts for LSF administrators (e.g., `lsfadmin`).
   - Read the “Release Notes for Platform LSF” (`/distrib/6.2/release_notes.html`) on the `ftp.platform.com` FTP site for detailed steps for downloading LSF distribution tar files
   - Get the LSF installation script tar file `lsf6.2_lsfinstall.tar.Z` and extract it. For example:

     ```
     # zcat lsf6.2_lsfinstall.tar.Z | tar xvf -
     ```
   - Read `lsf6.2_lsfinstall/README` for information about the contents of `lsf6.2_lsfinstall.tar.Z`.
   - Get the distribution tar files for all host types you need.

     Put the distribution files in the same directory as `lsf6.2_lsfinstall.tar.Z`

     Do not uncompress and extract the distribution tar files.

   - Get a valid license key and create a license file (`license.dat`) in the same directory as the distribution files and `lsf6.2_lsfinstall.tar.Z`.

If you do not specify a license file with LSF_LICENSE, or `lsfinstall` cannot find a license file in the default location, `lsfinstall` exits.

- Make sure the installation file system containing LSF_TOP is writable by the user account that is running `lsfinstall`.

Running lsfinstall

1. Log on as root to the installation file server.
2. Edit `lsf6.2_lsfinstall/install.config` or `lsf6.2_lsfinstall/slave.config`.

   Uncomment the options you want in the template file, and replace the example values with your own settings.

   To enable Platform LSF HPC installation, specifiy ENABLE_HPC_INST=Y in `install.config`.

   The sample values in the install.config and slave.config template files are examples only. They are *not* default installation values.

3. Change to `lsf6.2_lsfinstall/`.
4. Run `lsfinstall`:
   - `# ./lsfinstall -f install.config`

   OR

   - `# ./lsfinstall -s -f slave.config`
5. Before using your cluster, read the following:
   - `lsf6.2_lsfinstall/lsf_getting_started.html` to find out how to set up your LSF hosts, start LSF and test your new cluster.
   - `lsf6.2_lsfinstall/lsf_quick_admin.html` to learn more about your new cluster.

If you install as a non-root user

You can install as a non-root user with some limitations. During installation, `lsfinstall` detects that you are not root. You must choose to configure either a multi-user cluster or a single-user cluster:

- Single-user— Your user account must be primary LSF administrator. You can start LSF daemons, but only your user account can submit jobs to the cluster. Your user account must be able to read the system kernel information, such as `/dev/kmem`.
- Multi-user—By default, only root can start the LSF daemons. Any user can submit jobs to your cluster. To make the cluster available to other users, you must manually change the ownership and setuid bit for `lsadmin` and `badmin` to root, and the file permission mode to `-rwsr-xr-x` (4755) so that the user ID bit for the owner is `setuid`.

  Use the following commands to set the correct owner, user ID bit, and file permission mode for a multi-user cluster:

```
# chown root lsadmin badmin eauth
# chmod 4755 lsadmin badmin eauth
```

After installing Platform LSF

1 Optional. Run `hostsetup` to configure host-based resources and set up automatic LSF startup on your server hosts.

> For Platform LSF HPC hosts, running hostsetup is optional on AIX and Linux. You must run hostsetup on SGI IRIX, TRIX, and Altix hosts, and on HP-UX hosts.

a Log on to each server host as root. Start with the master host.

If you are not root, you can continue with host setup, but by default, only root can start the LSF daemons.

b Run `hostsetup` on each server host. For example:

```
# cd /usr/share/lsf/6.2/install
# ./hostsetup --top="/usr/share/lsf" --boot="y"
```

For complete `hostsetup` usage, enter `hostsetup -h`.

2 Log on to the LSF master host as root, and set your LSF environment:

- For `csh` or `tcsh`:

```
% source LSF_TOP/conf/cshrc.lsf
```

- For `sh`, `ksh`, or `bash`:

```
$ . LSF_TOP/conf/profile.lsf
```

3 Run `lsfstartup` to start the cluster.

> For large cluster, where cluster management software exists, you should ust /etc/init.d lsf start instead of lsftartup.

4 Test your cluster by running some basic commands (e.g., `lsid`, `lshosts`, `bhosts`)

After testing your cluster, be sure all LSF users include `LSF_CONFDIR/cshrc.lsf` or `LSF_CONFDIR/profile.lsf` in their `.cshrc` or `.profile`.

Follow the steps in `lsfinstall/lsf_quick_admin.html` for using `LSF_CONFDIR/cshrc.lsf` and `LSF_CONFDIR/profile.lsf` to set up the Platform LSF environment for users.

hostsetup example

The following example sets up a host to use the cluster installed in `/usr/share/lsf`. It also configures the LSF daemons to start automatically (`--boot="y"`):

```
# hostsetup --top="/usr/share/lsf" --boot="y"
```

Running host setup remotely (rhostsetup)

Use the `rhostsetup` script to launch `hostsetup` on remote hosts.

`rhostsetup` uses either `ssh` or `rsh`. It is included in the installation script tar file `lsf6.2_lsfinstall.tar.Z` and is located in the `lsf6.2_lsfinstall` directory created when you uncompress and extract installation script tar file.

After installation, `rhostsetup` is located in `LSF_TOP/6.2/install/`.

rhostsetup parameters

Before using `rhostsetup`, you must configure the following parameters at the top of the script:

- LSF_RSHCMD—the remote shell command (e.g, `rsh` or `ssh`) accessing the remote host
- LSF_HOSTS—list of hosts to run `hostsetup` on
- LSF_TOPDIR—sets the `hostsetup --top` option. Specify the full path to the top-level installation directory. `rhostsetup` tries to detect this from `lsf.conf` if it is not defined here.
- LSF_BOOT—sets the `hostsetup --boot` option. Default is no (`n`).
- LSF_QUIET—sets the `hostsetup --quiet` option. Default is no (`n`).

Example rshostsetup configuration

```
LSF_RSHCMD="ssh -n"
LSF_HOSTS="hostA hostB hostC"
LSF_TOPDIR=/usr/local/lsf
LSF_BOOT=y
LSF_QUIET=n
```

OPTIONS

-f *option_file*

Name of the file containing the installation options. The file can be any name 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.

Required parameters:

- LSF_SERVER_HOSTS="*host_name* [*host_name* ...]"
- LSF_TOP="/*path*"
- LSF_TARDIR="/*path*"

Optional parameters:

- 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, defined name-value pairs:

    "[resourcemap *value*__*__*resource_name*]"
  - For Boolean resources, the value will be the resource name in the form:

    "[resource *resource_name*]"

  For example:

```
LSF_LOCAL_RESOURCES="[resourcemap 1*verilog] [resource linux]"
```

If LSF_LOCAL_RESOURCES are already defined in a local `lsf.conf` on the slave host, `lsfinstall` does not add resources you define in LSF_LOCAL_RESOURCES in `slave.config`.

`lsfinstall` creates a local `lsf.conf` 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=6.2

-h

Prints command usage and exits.

SEE ALSO

`lsf.conf`(5), `install.config`(5), `slave.config`(5)

lsfmon

installs or uninstalls LSF Monitor

SYNOPSIS

`lsfmon -install`

`lsfmon -remove`

DESCRIPTION

Installs or uninstalls LSF Monitor in an existing cluster.

LSF Monitor runs on Microsoft Windows and allows you to use 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`

Installs LSF Monitor on the host.

`-remove`

Removes LSF Monitor from the host.

lsfrestart

restarts LIM, RES, `sbatchd` and `mbatchd` on all hosts in the cluster

SYNOPSIS

lsfrestart [**-f** | **-h** | **-V**]

DESCRIPTION

This command can only be used by root or users listed in lsf.sudoers.

Restarts LIM, RES, `sbatchd` and `mbatchd`, in that order, on all hosts in the local cluster.

By default, prompts for confirmation of the next operation if an error is encountered.

In order to be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly.
- You must be able to run the `rsh` or `ssh` command across all LSF hosts without having to enter a password. See your operating system documentation for information about configuring the `rsh` and `ssh` commands.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

OPTIONS

-f

Force mode. Continues to restart daemons even if an error is encountered.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

SEE ALSO

`lsadmin`(8), `badmin`(8), `lsfshutdown`(8), `lsf.sudoers`(5)

lsfsetcluster

specifies a default LSF cluster for the host

SYNOPSIS

lsfsetcluster *cluster_name*

lsfsetcluster [**-h** | **-V**]

DESCRIPTION

You must be a Windows local administrator of this host.

This command specifies the LSF cluster that users of the host interact with by default, and modifies LSF_BINDIR and LSF_ENVDIR system environment variables on the host.

Users of the host must set a different environment to interact with a different cluster.

OPTIONS

cluster_name

Specify an existing cluster. The host must already belong to the cluster (must have the appropriate LSF services and binary files installed, and must be listed in the cluster configuration file).

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

lsfshutdown

shuts down LIM, RES, `sbatchd` and `mbatchd` on all hosts in the cluster

SYNOPSIS

lsfshutdown [**-f** | **-h** | **-V**]

DESCRIPTION

This command can only be used by root or users listed in lsf.sudoers.

Shuts down `sbatchd`, RES, LIM, and `mbatchd`, in that order, on all hosts.

By default, prompts for confirmation of the next operation if an error is encountered.

In order to be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly.
- You must be able to run the `rsh` or `ssh` command across all LSF hosts without having to enter a password. See your operating system documentation for information about configuring the `rsh` and `ssh` commands.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

OPTIONS

-f

Force mode. Continues to shut down daemons even if an error is encountered.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

SEE ALSO

`lsadmin`(8), `badmin`(8), `lsfrestart`(8), `lsf.sudoers`(5)

lsfstartup

starts LIM, RES, `sbatchd`, and `mbatchd` on all hosts in the cluster

SYNOPSIS

lsfstartup [**-f**]

lsfstartup [**-h** | **-V**]

DESCRIPTION

This command can only be used by root or users listed in lsf.sudoers.

Starts LIM, RES, `sbatchd`, and `mbatchd`, in that order, on all hosts.

By default, prompts for confirmation of the next operation if an error is encountered.

If LSF daemons are already running, use `lsfrestart` instead, or use `lsfshutdown` and shut down the running daemons before you use `lsfstartup`.

In order to be able to control all daemons in the cluster:

- The file `/etc/lsf.sudoers` has to be set up properly.
- You must be able to run the `rsh` or `ssh` command across all LSF hosts without having to enter a password. See your operating system documentation for information about configuring the `rsh` and `ssh` commands.

The shell command specified by LSF_RSH in `lsf.conf` is used before `rsh` is tried.

OPTIONS

-f

Force mode. Continues to start daemons even if an error is encountered.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

SEE ALSO

`lsadmin`(8), `badmin`(8), `lsf.sudoers`(5), `lsfshutdown`(8), `lsfrestart`(8), `lsf.sudoers`(5)

lsgrun

executes a task on a set of hosts

SYNOPSIS

lsgrun [**-i**] [**-p** | **-P** | **-S**] [**-v**] **-f** *host_file* | **-m** *host_name* ... | **-n** *num_hosts*
[**-R** "*res_req*"] [*command* **[***argument* ...**]**]

lsgrun [**-h** | **-V**]

DESCRIPTION

Executes a task on the specified hosts. `lsgrun` is useful for fast global operations such as starting daemons, replicating files to or from local disks, looking for processes running on all hosts, checking who is logged in on each host, and so on. The hosts can be specified using a host file, a list of host names or by letting the system select the hosts.

DEFAULT BEHAVIOR

By default:

- `lsgrun` is not interactive.
- The specified task will be executed sequentially on hosts with full pseudo `tty` support.
- `lsgrun` 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 (task) terminated by a `CTRL-D` or `EOF`. The command is then executed on the specified hosts.

OPTIONS

-i

Interactive operation mode. You are asked whether the task will be executed on all hosts. If you answer `y`, the task is started on all specified hosts; otherwise, you are asked to specify hosts interactively.

-p

Parallel run mode. Executes the task on all hosts simultaneously and without pseudo `tty` support.

If this option is specified and the `-P` option is specified, the `-P` option is ignored.

This option is useful for fast start-up of tasks. However, any output from remote tasks will arrive at the terminal in arbitrary order, depending on task execution speeds on individual hosts.

-P

Creates a pseudo-terminal on UNIX hosts. This is necessary to run programs requiring a pseudo-terminal (for example, `vi`).

This option is not supported on Windows.

-S

Creates a pseudo-terminal with shell mode support on UNIX hosts.

Shell mode support is required for running interactive shells or applications which redefine the `CTRL-C` and `CTRL-Z keys` (such as `jove`).

This option is not supported on Windows.

-v

Verbose mode. Displays the name of the host or hosts running the task.

-f *host_file*

Either `-f` *host_file*, `-m` *host_name* or `-n` *num_processors* is required.

Executes the task on all hosts listed in the *host_file*.

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.

Executes the task on all specified hosts.

Specify hosts on which to execute the task. If multiple host names are specified, the host names must be enclosed by " or ' and separated by white space.

This option is exclusive of options `-n`, `-R`, and `-f`.

-n *num_hosts*

Either `-f` *host_file*, `-m` *host_name* or `-n` *num_hosts* is required.

Executes the task in a cluster with the required number of available hosts.

One host may be used to start several tasks if the host is multiprocessor. This option can be used together with option `-R` to select desired hosts.

This option is exclusive of options `-m` and `-f`.

-R "*res_req***"**

Executes the task on hosts with the required resource requirements.

Specify the resource requirement expression for host selection. The resource requirement will be used to choose from all hosts with the same host type as the local host, unless a "`type == value`" exists in *res_req* to specify otherwise.

This 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, you defined a resource called `bigmem` in `lsf.shared` and defined it as an exclusive resource for `hostE` in `lsf.cluster.mycluster`. Use the following command 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 execute. This 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 executed correctly.

Otherwise, the exit status is the first non-zero status returned by a remotely executed task. `lsgrun` will execute the task on all hosts even if some have non-zero exit status.

Exit status is -10 if a problem is detected in LSF.

SEE ALSO

`lsfintro`(1), `lsrun`(1), `lsplace`(1)

lshosts

displays hosts and their static resource information

SYNOPSIS

lshosts [**-w** | **-l**] [**-R "***res_req***"**] [*host_name* | *cluster_name*] ...

lshosts -s [*shared_resource_name* ...]

lshosts [**-h** | **-V**]

DESCRIPTION

Displays static resource information about hosts.

By default, returns the following information: host name, host type, host model, CPU factor, number of CPUs, total memory, total swap space, whether or not the host is a server host, and static resources. Displays information about all hosts in the cluster. See `lsf.cluster`(5).

In MultiCluster 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 or not 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. See `lsf.cluster`(5).

In MultiCluster resource leasing model, returns information about hosts in the local cluster.

The `-s` option displays information about the static shared resources and their associated hosts.

OPTIONS

-w

Displays host information in wide format. Fields are displayed without truncation.

-l

Displays host information in a long multi-line format. In addition to the default fields, displays information about the maximum `/tmp` space, the number of local disks, the execution priority for remote jobs, load thresholds, run windows, and the license classes used or needed.

If LSF_ENABLE_DUALCORE=Y in `lsf.conf` for dual-core CPU hosts, `lshosts -l` also displays the number of dual-core licenses enabled and needed.

-R "*res_req***"**

Only displays information about the hosts that satisfy the resource requirement expression. For more information about resource requirements, see `lsfintro`(1). The size of the resource requirement string is limited to 512 bytes. LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

In MultiCluster, only displays information about the hosts in the local cluster that satisfy the resource requirement expression.

host_name...| *cluster_name...*

Only displays information about the specified hosts. Do not use quotes when specifying multiple hosts.

For MultiCluster, displays information about hosts in the specified clusters. The names of the hosts belonging to the cluster are displayed instead of the name of the cluster. Do not use quotes when specifying multiple clusters.

-s [*shared_resource_name* ...]

Displays information about the specified resources. The resources must be static shared resources. If no shared resource is specified, then displays information about all shared 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.

-V

Prints the LSF release version to `stderr` and exits.

OUTPUT

Host-Based Default

Displays the following fields:

HOST_NAME

The name of the host. This display field is truncated.

type

The host type. This display field is truncated.

With MultiCluster, if the host type of a remote cluster's host is not defined in the local cluster, the keyword `unknown` will be displayed.

model

The host model. This display field is truncated.

With MultiCluster, if the host model of a remote cluster's host is not defined in the local cluster, the keyword `unknown` will be 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 CPU factor of a host with an `unknown` host type is 1.0.

ncpus

The number of processors on this host.

If LSF_ENABLE_DUALCORE=Y in `lsf.conf` for dual-core CPU hosts, displays the number of cores instead of physical CPUs

maxmem

The maximum amount of physical memory available for user processes.

maxswp

The total available swap space.

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 defined for this host, denoted by resource names, and the values of external numeric and string static resources. See `lsf.cluster`(5), and `lsf.shared`(5) on how to configure external static resources.

Host Based -l Option

In addition to the above fields, the `-l` option also displays the following:

ndisks

The number of local disk drives directly attached to the host.

maxtmp

The maximum `/tmp` space in megabytes configured on a host.

rexpri

UNIX only. The execution priority of remote jobs 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.

RUN_WINDOWS

The time windows during which LIM considers the host as available to execute remote jobs. These run windows have the same function for LSF hosts as dispatch windows have for LSF hosts. (See `lsf.cluster`(5).)

LICENSES_ENABLED

The licenses that are enabled for each specified host.

Also shows if dual-core CPU license is enabled for the hosts.

LICENSE CLASS NEEDED

The required banded license class for each specified host.

Also shows if dual-core CPU license is needed on the 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’s status. See `lsload`(1).

Resource-Based -s Option

Displays the static shared resources. Each line gives the value and the associated hosts for the static shared resource. See `lsf.shared`(5), and `lsf.cluster`(5) on how to configure static shared resources.

The following fields are displayed:

RESOURCE

The name of the resource.

VALUE

The value of the static shared resource.

LOCATION

The hosts that are associated with the static shared resource.

FILES

Reads `lsf.cluster.`*`cluster_name`*.

SEE ALSO

`lsfintro`(1), `ls_info`(3), `ls_policy`(3), `ls_gethostinfo`(3), `lsf.cluster`(5), `lsf.shared`(5)

lsid

displays the current LSF version number, the cluster name, and the master host name

SYNOPSIS

lsid [**-h** | **-V**]

DESCRIPTION

Displays the current LSF version number, the cluster name, and the master host name.

The master host is dynamically selected from all hosts in the cluster.

OPTIONS

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

FILES

The host names and cluster names are defined in `lsf.cluster.`*cluster_name* and `lsf.shared`, respectively.

SEE ALSO

`ls_getclustername`(3), `ls_getmastername`(3), `lsinfo`(1)

lsinfo

displays load sharing configuration information

SYNOPSIS

lsinfo [**-l**] [**-m** | **-M**] [**-r**] [**-t**] [*resource_name* ...]

lsinfo [**-h** | **-V**]

DESCRIPTION

By default, displays all load sharing 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 only information about host models that exist in the cluster.

-M

Displays information about all host models in the file `lsf.shared`.

-r

Displays only information about configured resources.

-t

Displays only information about configured host types. See `lsload(1)` and `lshosts(1)`.

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

-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, such as floating licenses. 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 `lsf.shared`.

`No` indicates the resource is held. `Yes` indicates the resource is released.

SEE ALSO

`lsfintro`(1), `lshosts`(1), `lsload`(1), `lsf.shared`(5), `ls_info`(3), `ls_policy`(3)

lsload

displays load information for hosts

SYNOPSIS

lsload [**-l**] [**-N** | **-E**] [**-I** *load_index*[**:***load_index*] ...] [**-n** *num_hosts*] [**-R** *res_req*]
[*host_name* ... | *cluster_name* ...]

lsload -s [*resource_name* ...]

lsload [**-h** | **-V**]

DESCRIPTION

Displays load information for hosts. 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, per host.

With MultiCluster, also displays load information for all hosts in equivalent clusters (see `lsf.cluster(5)`).

By default, displays raw load indices.

By default, load information for resources is displayed according to CPU and paging load.

OPTIONS

-l

Long format. Displays load information without truncation along with additional fields for I/O and external load indices.

This option overrides the index names specified with the `-I` option.

-N

Displays normalized CPU run queue length load indices (see `lsfintro(1)`).

-E

Displays effective CPU run queue length load indices (see `lsfintro(1)`). Options `-N` and `-E` are mutually exclusive.

-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.

-R *res_req*

Displays only load information for hosts that satisfy the specified resource requirements. See `lsinfo(1)` for a list of built-in resource names.

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 (see `lshosts(1)` 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 MultiCluster, 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 MultiCluster, displays only load information for hosts in the specified clusters.

-s [*resource_name ...*]

Displays information about all dynamic shared resources configured in the cluster, or about the specified resources only. Specify dynamic shared resources.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

OUTPUT

HOST-BASED OUTPUT (default 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* (see `lsf.cluster(5)`). The selection and order sections of *res_req* control for which hosts are displayed and how the information is ordered (see `lsfintro(1)`).

The display includes the following fields:

HOST_NAME

Standard host name used by LSF, typically an Internet domain name with two components.

status

Status of the host. A minus sign (-) may precede the status, indicating that RES is not running on the host.

Possible statuses are:

ok

The host is in normal load sharing state and can accept remote jobs.

-ok

The Load Information Manager (LIM) on the host is running but the Remote Execution Server (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 configuration file (see `lsf.conf(5)`) and can be displayed by `lshosts`. A locked host will 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 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, between 0 and 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 (keyboard not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in `/tmp`, in megabytes.

swp

The amount of available swap space, in megabytes.

mem

The amount of available RAM, in megabytes.

io

By default, `io` is not shown.

If -l is specified, shows the disk I/O rate exponentially averaged over the last minute, in KB per second.

external_index

By default, external load indices are not shown.

If -l is specified, shows indices for all dynamic custom resources available on the host, including shared, string and Boolean resources.

If -I *load_index* is specified, only shows indices for specified non-shared (host-based) dynamic numeric custom resources.

RESOURCE-BASED OUTPUT (lsload -s)

Displays information about dynamic shared resources. Each line gives the value and the associated hosts for an instance of the resource. See `lim(8)`, and `lsf.cluster(5)` for information on configuring dynamic shared resources.

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.

EXAMPLES

```
% lsload -R "select[r1m<=0.5 && swp>=20 && type==ALPHA]"
```

OR, in restricted format:

```
% lsload -R r1m=0.5:swp=20:type=ALPHA
```

Displays the load of ALPHA hosts with at least 20 megabytes of swap space, and a 1-minute run queue length less than 0.5.

```
% lsload -R "select[(1-swp/maxswp)<0.75] order[pg]"
```

Displays the load of the hosts whose swap space utilization is less than 75%. The resulting hosts are ordered by paging rate.

```
% lsload -I r1m:ut:io:pg
```

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 -E
```

Displays the load of all hosts, ordered by `r15s:pg`, with the CPU run queue lengths being the effective run queue lengths (see `lsfintro(1)`).

```
% lsload -s verilog_license
```

Displays the value and location of all the `verilog_license` dynamic shared resource instances.

DIAGNOSTICS

Exit status is -10 for LSF problems or a bad resource names.

Exit status is -1 if a bad parameter is specified, otherwise `lsload` returns 0.

SEE ALSO

`lsfintro`(1), `lim`(8), `lsf.cluster`(5), `lsplace`(1), `lshosts`(1), `lsinfo`(1), `lslockhost`(8), `ls_load`(3)

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.

`lsloadadj` 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%

OPTIONS

-R *res_req*

Specify resource requirements for tasks. Only the resource usage section of the resource requirement string is considered (see `lsfintro(1)`). 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 a bad parameter is specified; otherwise returns 0.

SEE ALSO

`lsinfo`(1), `lsplace`(1), `lsload`(1), `ls_loadadj`(3)

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

Remotely logs in to a lightly loaded host.

By default, `lslogin` selects the least loaded host, with few users logged in, and remotely logs in to that host using the UNIX `rlogin` command.

In a MultiCluster environment, the default is to select the least loaded host in the local cluster.

OPTIONS

-v

Displays the name of the host to which `lslogin` remotely logs you in.

-m "*host_name ...***"** | **-m "***cluster_name ...***"**

Remotely logs in to the specified host.

With MultiCluster job forwarding, when a cluster name is specified, remotely logs in to the least loaded host in the specified cluster, if the remote cluster accepts interactive jobs from the local cluster (see `lsf.cluster(5)`).

-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.

For a complete explanation of resource requirement expressions, see `lsfintro(1)`. To find out what resources are configured in your system, use `lsinfo(1)` and `lshosts(1)`.

rlogin_options

Specify remote login options passed to the `rlogin` command.

If remote execution fails, `lslogin` will log in locally only if the local host also satisfies required resources; otherwise, log in will fail.

-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 has been idle for at least 1 minute, that runs BSD UNIX, and is lightly loaded both in CPU resources and the number of users logged in.

DIAGNOSTICS

Because `lslogin` passes all unrecognized arguments to `rlogin`, incorrect options usually cause the `rlogin` usage message to be displayed rather than the `lslogin` usage message.

SEE ALSO

`lsfintro`(1), `ls_placereq`(3), `rlogin`(1)

lsltasks

displays or updates a user's local task list

SYNOPSIS

lsltasks [**+** *task_name* ... | – *task_name* ...]

lsltasks [**-h** | **-V**]

DESCRIPTION

Displays or updates a user's local task list in `$HOME/.lsftask`.

When no options are specified, displays tasks listed in the system task file `lsf.task` and the user's task file `.lsftask`.

If there is a conflict between the system task file `lsf.task` and the user's task file `.lsftask`, the user's 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 + is specified and the specified task names are not already in the file `.lsftask` in the user's home directory, 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 – is specified and specified task names are not already in the file `.lsftask` in the user's home directory, 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 task file `.lsftask` in the user's home directory. See `lsf.task(5)` for more details.

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 an entry consisting 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

`lsfintro`(1), `lseligible`(1), `ls_task`(3), `lsrtasks`(1), `lsf.task`(5), `ls_eligible`(3)

lsmake

runs make tasks in parallel

SYNOPSIS

lsmake [**-c** *num_tasks*] [**-F** *res_req*] [**-m "***host_name* ...**"**] [**-E**] [**-G**] [**-M**] [**-V**] [*makeoption* ...] [*target* ...]

lsmake [**-c** *num_tasks*] [**-F** *res_req*] [**-T**] [**-j** *max_processors*] [**-P** *minutes*] [**-R** *res_req*] [**-E**] [**-G**] [**-M**] [**-V**] [*makeoption* ...] [*target* ...]

DESCRIPTION

Runs make tasks in parallel on LSF hosts. Sets the environment variables on the remote hosts when `lsmake` first starts.

By default, uses the local host, uses only one processor, starts only one task in each processor, and processes submakes sequentially.

`lsmake` is a modified version of `GNU make`. All the options provided by `GNU make` are valid with `lsmake`.

OPTIONS

-E

Sets the environment variables for every task sent remotely.

This is necessary when make files change or override the environment variables they inherit at startup.

-G

Enables debugging.

-M

Processes submakes in parallel. Some makefiles may not work correctly when run in parallel through Platform Make.

To use this feature, build each submake as a separate target in your makefile. Specify the make command for each submake with the built-in $(MAKE) macro. Makefiles that depend on sequential processing might have to be modified further.

For more information, see the Platform Make documentation.

-T

Enables output tagging to prefix the sender's task ID to the parallel task output data.

-V

Verbose mode. Prints the names of the hosts used.

-c *num_tasks*

Starts the specified number of tasks concurrently on each processor. If you specify too many tasks, you could overload a host.

-F *res_req*

Temporarily reduces the number of tasks running when the load on the network file server exceeds the specified resource requirements. This might also reduce the number of processors used. The number of tasks is increased again when the load on the network file server is below the specified resource requirements.

The network file server is considered to be the host mounting the current working directory on the local host. If this machine is not in the local cluster, `-F` is ignored.

-m "*host_name ...***"**

Uses the specified hosts. Specify a host name multiple times to use multiple processors on that host.

-j *max_processors*

Uses multiple processors. Specify the maximum number of processors to use. Uses all of the available processors if fewer processors are available.

When you specify `-j` and `-R` together, automatically selects processors on the best available hosts that satisfy the resource requirements. The job fails if no suitable host is found.

When you specify `-j` but not `-R`, automatically selects processors on the best available hosts that are the same host type as the local host. The local host itself can be selected.

-P *minutes*

Periodically reselects the best available processors. After the processor has been used for the specified number of minutes, it might be replaced if a better processor is available.

This is useful for long-running makes.

-R *res_req*

Uses only hosts that satisfy the specified resource requirements.

When you specify `-R` but not `-j`, uses one processor on one host that satisfies the resource requirements.

makeoption ...

Specifies `GNU Make` options. See `gmake(1)` for details.

target ...

Specifies targets to make.

LIMITATIONS

If a submake in a makefile specifies options which are specific to `lsmake`, they are ignored. Only the command line options are used. When determining where to start tasks, `lsmake` consults the local task list (see `lsf.task`(5)). If the task is found in the local task list, it will be started on the local host. The resource requirements of tasks in the remote task list are not considered when dispatching tasks.

SEE ALSO

`lsfintro`(1), `lstcsh`(1), `gmake`(1)

For a complete description of how to use Platform LSF Make, see the Platform LSF Make documentation.

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

`lsmon` 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 will 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 (see `lsfintro(1)`).

-E

Displays effective CPU run queue length load indices (see `lsfintro(1)`). 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. See `lsinfo`(1) for a list of built-in resource names.

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 (see `lshosts(1)` 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 of the user who invoked the command, 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 `lim.acct`(5).

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 (-) may precede the status, indicating that the Remote Execution Server (RES) on the host is not running.

Possible statuses are:

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` (see below). External load indices are configured in the file `lsf.cluster.`*cluster_name* (see `lsf.cluster(5)`).

lockW

The host is locked by its run window. Run windows for a host are specified in the configuration file (see `lsf.conf(5)`) and can be displayed by `lshosts`. A locked host will 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, between 0 and 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 (keyboard not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in `/tmp`, in megabytes.

swp

The amount of currently available swap space, in megabytes.

mem

The amount of currently available memory, in megabytes.

DIAGNOSTICS

Specifying an incorrect resource requirement string while `lsmon` is running (via the `R` option) causes `lsmon` to exit with an appropriate error message.

`lsmon` exits if it does not receive a reply from LIM within the update interval.

SEE ALSO

`lsfintro`(1), `lshosts`(1), `lsinfo`(1), `lsload`(1), `lslockhost`(8), `lim.acct`(5), `ls_load`(3)

lspasswd

registers user passwords in LSF on Windows

SYNOPSIS

lspasswd [**-u** *user_name*]

DESCRIPTION

Registers user passwords in LSF on Windows. Passwords must be 3 characters or longer.

By default, if no options are specified, the password applies to the user who issued the command.

Only the LSF administrator can enter passwords for other users.

Users must update the password 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.

`lspasswd` communicates with LSF services on the local machine to store the password. The password is stored in encrypted format and the password database is protected by Windows file access permissions.

OPTIONS

-u user_name

Specify the user whose password you want to change. Only the LSF administrator can enter passwords for other users.

LIMITATIONS

You must run `lspasswd` from an LSF server host. You cannot run the command from an LSF client host.

lsplace

displays hosts available to execute 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 will decay slowly over time before the real load produced by the dispatched task is reflected in the LIM's load information. Host names may 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. This is useful for distributed parallel applications in order 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 holding the required resources cannot be found.

-R *res_req*

Displays only hosts with the specified resource requirements.

-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

`lsplace` is mostly used in backquotes to pick out a host name which 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` and `-n` options can be combined to specify the upper and lower bounds in processors to be returned, respectively. For example, the command `lsplace -n 3 -w 5` returns at least 3 and not more than 5 host names.

DIAGNOSTICS

`lsplace` returns 1 if insufficient hosts are available. The exit status is -10 if a problem is detected in LSF, -1 for other errors, otherwise 0.

SEE ALSO

`lsinfo`(1), `ls_placereq`(3), `lsload`(1), `lsrun`(1)

lsrcp

remotely copies files using LSF

SYNOPSIS

lsrcp [**-a**] *source_file target_file*

lsrcp [**-h** | **-V**]

DESCRIPTION

Remotely copies files using LSF.

`lsrcp` is an LSF-enabled remote copy program that transfers a single file between hosts in an LSF cluster. `lsrcp` 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.

To use `lsrcp`, you must have read access to the file being copied.

Both the source and target file must be owned by the user who issues the command.

`lsrcp` uses `rcp` to copy a source file to a target file owned by another user. See `rcp(1)` and LIMITATIONS below for details.

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.

File format is as follows:

[[*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 issued the command is used.

host_name

Name of the remote host on which the file resides. If *host_name* is not specified, the local host, the host from which the command was issued 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 "\" to transfer files from a Windows host to another Windows host. For example:

`c:\share>`**`lsrcp file1 hostA:c:\temp\file2`**

Use "/" to transfer files from a UNIX host to a UNIX host. For example:

`% `**`lsrcp file1 hostD:/home/usr2/test/file2`**

Always use "`/`" to transfer files from a UNIX host to a Windows host, or from a Windows host to a UNIX host. This is because the operating system interprets "`\`" and `lsrcp` will open the wrong files.

For example, to transfer a file from UNIX to a Windows host:

`% `**`lsrcp file1 hostA:/c:/temp/file2`**

For example, to transfer 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.

`-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 `user1`'s home directory.

DIAGNOSTICS

`lsrcp` attempts to copy *source_file* to *target_file* using RES. If RES is down or fails to copy the *source_file*, `lsrcp` will use either `rsh` or the shell command specified by LSF_RSH in `lsf.conf` when the `-a` option is specified. When `-a` is not specified, `lsrcp` will use `rcp`.

LIMITATIONS

File transfer using `lscrp` is not supported in the following contexts:

- If LSF account mapping is used; `lsrcp` fails when running under a different user account
- On LSF client hosts. LSF client hosts do not run RES, so `lsrcp` cannot contact RES on the submission host
- Third party copies. `lsrcp` does not support third party copies, when neither source nor target file are on the local host. In such a case `rcp` or `rsh` (or the shell command specified by LSF_RSH in `lsf.conf`) will be used. If the *target_file* exists, `lsrcp` preserves the modes; otherwise, `lsrcp` uses the *source_file* modes modified with the `umask` (see `umask(2)`) of the source host.

You can do the following:

- rcp on UNIX. If lsrcp cannot contact RES on the submission host, it attempts to use rcp to copy the file. You must set up the /etc/hosts.equiv or HOME/.rhosts file in order to use rcp. See the rcp(1), rsh(1), ssh(1) manual pages for more information on using the rcp, rsh, and ssh commands.
- You can replace lsrcp with your own file transfer mechanism as long as it supports the same syntax as lsrcp. This might be done to take advantage of a faster interconnection network, or to overcome limitations with the existing lsrcp. sbatchd looks for the lsrcp executable in the LSF_BINDIR directory as specified in cshrc.lsf, profile.lsf, or lsf.conf.

SEE ALSO

rsh(1), rcp(1), lsfintro(1), res(8)

lsrtasks

displays or updates a user's 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 `$HOME/.lsftask`.

When no options are specified, displays tasks listed in the system task file `lsf.task` and the user's task file `.lsftask`.

If there is a conflict between the system task file `lsf.task` and the user's task file `.lsftask`, the user's 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 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 when deciding the eligibility of a task (see `lseligible(1)`, and `ls_eligible(3)`).

OPTIONS

+ *task_name*[**/***res_req*] ...

If plus sign (+) is specified and the specified task names are not already in the file `.lsftask` in the user's home directory, adds the task names to the file with a + 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.

Remote tasks can have a resource requirement expression associated with them, separated by a backslash (/). See `ls_task(3)`.

- *task_name*[**/***res_req*] ...

If – is specified and specified task names are not already in the file `.lsftask` in the user's home directory, 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 a resource requirement expression associated with them, separated by a backslash /. See `ls_task`(3).

-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, 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` in the user's home directory. See `lsf.task`(5) for more details.

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 consisting 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

`lsfintro`(1), `lseligible`(1), `ls_task`(3), `lsltasks`(1), `lsf.task`(5), `ls_eligible`(3)

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

Submits a task to LSF for execution.

With MultiCluster job forwarding model, the default is to run the task on a host in the local cluster.

By default, `lsrun` first tries to obtain resource requirement information from the remote task list to find an eligible host. (See `lseligible`(1) and `ls_task`(3).) Otherwise, `lsrun` 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, `lsrun` does not create a pseudo-terminal when running the task.

OPTIONS

-l

If execution on another host fails, runs the task locally.

-L

Forces `lsrun` to go through RES to execute a task. By default, `lsrun` will not use RES if the task is going to run on the current host.

-P

Creates a pseudo-terminal when starting the task on UNIX hosts. This is necessary in order to run programs that require a pseudo-terminal (for example, `vi`).

This option is not supported on Windows.

-S

Creates a pseudo-terminal with shell mode support when starting the task on a UNIX host. Shell mode support is required for running interactive shells or applications which 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 running 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 `lsrtasks`(1)). If none of the specified hosts satisfy the resource requirements, the task will not run.

With MultiCluster 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. (See `RemoteClusters` section in `lsf.cluster`(5).)

-R "*res_req*"

Runs the task on a host that meets the specified resource requirement. The size of the resource requirement string is limited to 512 bytes. For a complete explanation of resource requirement expressions, see `lsfintro`(1). To find out what resources are configured in your system, use `lsinfo`(1) and `lshosts`(1).

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 called `bigmem` in `lsf.shared` and defined it as an exclusive resource for `hostE` in `lsf.cluster.mycluster`. Use the following command 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 `lsrun` together with other utility commands such as `lsplace`(1), `lsload`(1), `lsloadadj`(1), and `lseligible`(1) to write load sharing applications in the form of UNIX shell scripts.

`lsrun` supports interactive job control. Suspending `lsrun` suspends both the task and `lsrun`, and continuing `lsrun` continues the task.

The `-n` option of `rsh`(1) can be simulated by redirecting input from `/dev/null`. For example:

```
lsrun cat </dev/null &
```

DIAGNOSTICS

lsrun exits with status -10 and prints an error message to stderr if a problem is detected in LSF and the task is not run.

The exit status is -1 and an error message is printed to stderr if a system call fails or incorrect arguments are specified.

Otherwise, the exit status is the exit status of the task.

SEE ALSO

rsh(1), lsfintro(1), ls_rexecv(3), lsplace(1), lseligible(1), lsload(1), lshosts(1), lsrtasks(1), lsf.cluster(5)

lstcsh

load sharing `tcsh` for LSF

SYNOPSIS

lstcsh [*tcsh_options*] [**-L**] [*argument* ...]

DESCRIPTION

`lstcsh` is an enhanced version of `tcsh`. `lstcsh` behaves exactly like `tcsh`, except that it includes a load sharing capability with transparent remote job execution for LSF.

By default, a `lstcsh` script is executed as a normal `tcsh` script with load sharing disabled.

If a command line is considered eligible for remote execution, LSF selects a suitable host— typically a powerful and/or lightly loaded host that can execute the command line correctly—and sends the command line to that host.

You can restrict who can use `@` for host redirection in `lstcsh` with the parameter LSF_SHELL_AT_USERS in `lsf.conf`.

Remote Hosts

`lstcsh` provides a high degree of network transparency. Command lines executed on remote hosts behave the same as they do on the local host. The remote execution environment is designed to mirror the local one as closely as possible by using the same values for environment variables, terminal setup, current working directory, file creation mask, and so on. 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 `lstcsh` is exactly the same as in `tcsh` except for remote background jobs. `lstcsh` 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 killing a remote job. For example:

```
kill %2 @hostA
```

OPTIONS

`tcsh_options`

`lstcsh` accepts all the options used by `tcsh`. See `tcsh(1)` for the meaning of specific options.

`-L`

Executes a script with load sharing enabled.

There are three ways to run a `lstcsh` script with load sharing enabled:

- Execute the script with the `-L` option

- Use the built-in command `source` to execute the script

- Insert "`#!/local/bin/lstcsh -L`" as the first line of the script (assuming you install `lstcsh` in `/local/bin`).

Using `@` or `lsmode` (see below) in a script will not enable load sharing if the script has not been executed using one of these three ways.

USAGE

In addition to the built-in commands in `tcsh`, `lstcsh` 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 send a command line to a remote host only if forced eligibility is specified with `@`.

`local | remote`

Sets operation mode of `lstcsh`.

The default is `local`.

`local`

Local operation mode. This is the default mode.

In this mode, a command line 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 line to force specified tasks to be eligible for remote execution.

Tasks in the local task list must be executed locally.

The local mode of operation is conservative, and can fail to take advantage of the performance benefits and load balancing advantages of LSF.

The way `lstcsh` handles tasks that are not present in the remote task list nor in the local task list, depends on the mode of operation of `lstcsh` (local or remote).

remote

Remote operation mode.

In this mode, a command line 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 executed remotely.

The remote mode of operation is aggressive, and promotes extensive use of LSF.

The way `lstcsh` handles tasks that are not present in the remote task list nor in the local task list, depends on the mode of operation of `lstcsh` (local or remote).

@

Specify `@` to explicitly specify the eligibility of a command for remote execution.

The `@` may be anywhere in the command line except in the first position (which is used to set the value of shell variables).

There are several ways to use `@`:

@

Specify `@` followed by nothing to indicate the command line is eligible for remote execution.

@ *host_name*

Specify `@` followed by a host name to force the command line to be executed on that host.

Host names and the reserved word `local` following `@` can all be abbreviated as long as they do not cause ambiguity.

@ local

Specify `@` followed by the reserved word `local` to force the command line to executed on the local host.

@ /*res_req*

Specify `@` followed by `/` and a resource requirement string to indicate the command is eligible for remote execution, and that the specified resource requirements must be used instead of those in the remote task list.

When specifying resource requirements following the `@` it is necessary to use `/` only if the first requirement characters specified are also the first characters of a host name.

e | -e

Turns eligibility verbose mode on (`e`) or off (`-e`).

If eligibility verbose mode is on, `lstcsh` shows whether the command is eligible for remote execution, and displays the resource requirement used if the command is eligible.

The default is off.

v | -v

Turns task placement verbose mode on (`v`) or off (`-v`). If verbose mode is on, `lstcsh` displays the name of the host on which the command is run if the command is not run on the local host.

The default is on.

t | -t

Turns wall clock timing on (`t`) or off (`-t`).

If timing is on, the actual response time of the command is displayed. This 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 overhead. The `csh` time built-in does not include the remote execution overhead.

This is an impartial way of comparing the response time of jobs submitted locally or remotely, because all the load sharing overhead 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 has been established.

A plus sign (+) with a remote host indicates that a server-shell has also been 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`.

This 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 `lsrtasks(1)` 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`.

This 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 `lsltasks`(1) for more details.

jobs

Lists background jobs together with the execution hosts. This break of transparency is intentional in order to provide you with more control over your background jobs.

FILES

There are three optional configuration files for `lstcsh`:

`.shrc`

`.hostrc`

`.lsftask`

The `.shrc` and `.hostrc` files are used by `lstcsh` alone, whereas `.lsftask` is used by LSF to determine general task eligibility.

~/.shrc

Use this file when you want an execution environment on remote hosts that is different from that 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 may need to run a version of the executable for that particular host type, therefore it may 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 saves the time spent in establishing the connections dynamically during execution of shell commands. Once a connection is set up, you can execute further remote commands on those connected hosts with very little overhead.

~/.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 specifying the resource requirements of the task. If *res_req* is not specified, the command is executed on machines of the same type as the local host.

LIMITATIONS

Type-ahead for the `next` command is discarded when a job is executing in the foreground on a remote host.

It is not possible to provide input data to load sharing shell scripts (that is, shell scripts whose content is load shared).

The `lstcsh` is fully compatible with tcsh 6.03 7-bit mode. Any feature that is not included in `tcsh` 6.03 is not supported.

SEE ALSO

`csh`(1), `tcsh`(1), `lsrtasks`(1), `lsltasks`(1), `lseligible`(1), `lsinfo`(1), `lsload`(1)

pam

Parallel Application Manager – job starter for MPI applications

SYNOPSIS

HP-UX vendor MPI syntax

bsub pam -mpi mpirun [*mpirun_options*] *mpi_app* [*argument* ...]

SGI vendor MPI syntax

bsub pam [**-n** *num_tasks*] **-mpi -auto_place** *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**] [**-V**]

DESCRIPTION

The Parallel Application Manager (PAM) is the point of control for Platform LSF HPC. PAM is fully integrated with Platform LSF HPC to interface the user application with LSF. PAM acts as the supervisor of a parallel LSF job.

MPI jobs started by `pam` can only be submitted through the LSF Batch system. PAM cannot be used interactively to start parallel jobs. `sbatchd` starts PAM on the first execution host.

For all parallel application processes (tasks), PAM:

- Uses a vendor MPI library or an MPI Parallel Job Launcher (PJL; for example, `mpirun`, `poe`) to start a parallel job on a specified set of hosts in a LSF cluster.
- PAM contacts RES on each execution host 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 a LSF cluster. Using `pam` to start an MPI job requires the underlying MPI system to be LSF aware, using a vendor MPI implementation that supports LSF (SGI IRIX vendor MPI or HP-UX vendor MPI).

PAM uses the vendor MPI library to spawn the child processes needed for the parallel tasks that make up your MPI application. It starts these tasks on the systems 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 LSF HPC GENERIC PJL JOBS

For parallel jobs submitted with `bsub`:

- PAM invokes the PJL, which in turn invokes the TaskStarter (TS).
- TS starts the tasks on each execution host, reports the process ID to PAM, and waits for the task to finish.

OPTIONS

OPTIONS FOR VENDOR MPI JOBS

-auto_place

The `-auto_place` option on the `pam` command line tells the SGI IRIX `mpirun` library to launch the MPI application according to the resources allocated by LSF.

-mpi

In the SGI environment, the `-mpi` option on the `bsub` and `pam` command line is equivalent to the `mpirun` command.

On HP-UX, you can have LSF manage the allocation of hosts to achieve better resource utilization by coordinating the start-up phase with `mpirun`. This is done by preceding the regular HP MPI `mpirun` command with:

```
% bsub pam -mpi
```

For HP-UX vendor MPI jobs, the `-mpi` option must be the first option of the `pam` command.

For example, to run a single-host job and have LSF select the host, the command:

```
% mpirun -np 14 a.out
```

is entered as:

```
% bsub pam -mpi mpirun -np 14 a.out
```

-n *num_tasks*

The number of processors 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 `bsub -n` and `pam -n` in the same job submission. The number specified in the `pam -n` option should be less than or equal to the number specified by `bsub -n`. If the number of tasks specified with `pam -n` is greater than the number specified by `bsub -n`, the `pam -n` is ignored.

For example, on SGI IRIX or SGI Altix, you can specify:

```
% bsub -n 5 pam -n 2 -mpi -auto_place a.out
```

Here, the job requests 5 processors, but PAM only starts 2 parallel tasks.

mpi_app [argument ...]

The name of the MPI application to be run on the listed hosts. This 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 LSF HPC GENERIC PJL JOBS

-t

This option tells `pam` not to print out the MPI job tasks summary report to the standard output. By default, the summary report prints out the task ID, the host on which it was executed, the command that was executed, 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 LSF HPC 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 launches 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. The syntax is:

`pam [pam_options] -g 4 3_arg_pjl -a -b group_name job [job_options]`

-n num_tasks

The number of processors 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 `bsub -n` and `pam -n` in the same job submission. The number specified in the `pam -n` option should be less than or equal to the number specified by `bsub -n`. If the number of tasks specified with `pam -n` is greater than the number specified by `bsub -n`, the `pam -n` is ignored.

mpi_app [*argument* ...]

The name of the MPI application to be run on the listed hosts. This 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.

EXIT STATUS

`pam` exits with the exit status of `mpirun` or the PJL wrapper.

SEE ALSO

`bsub`(1)

taskman

checks out a license token and manages interactive UNIX applications

SYNOPSIS

taskman -Lp *project* **-R "rusage[***token***=***number*[**:duration=***minutes* | *hours***h**]
[**:***token***=***number*[**:duration=***minutes* | *hours***h**]]...**]** [**-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 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 License Scheduler. When the application terminates, 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 if you want unused licenses to be reallocated if the application fails to check out the license before the reservation expires.

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 taskman will take to connect to the daemon. If this option is not specified, taskman will retry indefinitely.

-Lp *project*

Required. Specifies the interactive license project that is requesting tokens. The client must be known to LSF License Scheduler.

-R "rusage[token=*number*[**:duration=***minutes* | *hours***h**][**:token=***number*[**:duration=***minutes* | *hours***h**]**]**...]

Required. Specifies the type and number of license tokens to request from GLB. 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 may specify multiple license types, with different duration values. Separate each requirement with a colon (:). Enclose the entire list in one set of square brackets.

-h

Prints command usage to `stderr` and exits.

-V

Prints the License Scheduler release version to `stderr` and exits.

wgpasswd

changes a user's password for an entire Microsoft Windows workgroup

SYNOPSIS

wgpasswd [*user_name*]

wgpasswd [**-h**]

DESCRIPTION

You must run this command on a host in a 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.

wguser

modifies user accounts for an entire Microsoft Windows workgroup

SYNOPSIS

wguser [**-r**] *user_name* ...

wguser [**-h**]

DESCRIPTION

You must run this command on a host in a Microsoft Windows workgroup. You should 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 already 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).

PART II

Environment Variables

Environment Variables

Contents

- "Environment Variables Set for Job Execution" on page 268
- "Environment Variable Reference" on page 269

Environment Variables Set for Job Execution

LSF transfers most environment variables between submission and execution hosts. In addition to environment variables inherited from the user environment, LSF also sets several other environment variables for batch jobs:

- LSB_ERRORFILE: Name of the error file specified with a `bsub -e`
- LSB_JOBID: Batch job ID assigned by LSF.
- LSB_JOBINDEX: Index of the job that belongs to a job array.
- LSB_CHKPNT_DIR: This variable is set each time a checkpointed job is submitted. The value of the variable is *chkpnt_dir/job_Id*, a subdirectory of the checkpoint directory that is specified when the job is submitted. The subdirectory is identified by the job ID of the submitted job.
- LSB_HOSTS: The list of hosts that are used to run the batch job.

  For sequential jobs, this is only one host name. For parallel jobs, this includes multiple host names.
- LSB_QUEUE: The name of the queue the job is dispatched from.
- LSB_JOBNAME: Name of the job.
- LSB_RESTART: Set to 'Y' if the job is a restarted job or if the job has been migrated. Otherwise this variable is not defined.
- LSB_EXIT_PRE_ABORT: Set to an integer value representing an exit status. A pre-execution command should exit with this value if it wants the job to be aborted instead of requeued or executed.
- LSB_EXIT_REQUEUE: Set to the REQUEUE_EXIT_VALUES parameter of the queue. This variable is not defined if REQUEUE_EXIT_VALUES is not configured for the queue.
- LSB_JOB_STARTER: Set to the value of the job starter if a job starter is defined for the queue.
- LSB_INTERACTIVE: Set to 'Y' if the job is submitted with the `-I` option. Otherwise, it is undefined.
- LS_JOBPID: Set to the process ID of the job.
- LS_SUBCWD: This is the directory on the submission when the job was submitted. This is different from PWD only if the directory is not shared across machines or when the execution account is different from the submission account as a result of account mapping.

Environment Variable Reference

BSUB_BLOCK	BSUB_QUIET	BSUB_QUIET2
BSUB_STDERR	CLEARCASE_DRIVE	CLEARCASE_MOUNTDIR
CLEARCASE_ROOT	LM_LICENSE_FILE	LS_EXEC_T
LS_JOBPID	LS_LICENSE_SERVER_*feature*	LS_SUBCWD
LSB_CHKPNT_DIR	LSB_DEBUG	LSB_DEBUG_CMD
LSB_DEBUG_MBD	LSB_DEBUG_NQS	LSB_DEBUG_SBD
LSB_DEBUG_SCH	LSB_DEFAULTPROJECT	LSB_DEFAULTQUEUE
LSB_ECHKPNT_METHOD	LSB_ECHKPNT_METHOD_DIR	LSB_ECHKPNT_KEEP_OUTPUT
LSB_ERESTART_USRCMD	LSB_EXEC_RUSAGE	LSB_EXECHOSTS
LSB_EXIT_PRE_ABORT	LSB_EXIT_REQUEUE	LSB_FRAMES
LSB_HOSTS	LSB_INTERACTIVE	LSB_JOB_STARTER
LSB_JOBEXIT_INFO	LSB_JOBEXIT_STAT	LSB_JOBFILENAME
LSB_JOBID	LSB_JOBINDEX	LSB_JOBINDEX_STEP
LSB_JOBNAME	LSB_JOBPEND	LSB_JOBPGIDS
LSB_JOBPIDS	LSB_MAILSIZE	LSB_MCPU_HOSTS
LSB_NQS_PORT	LSB_NTRIES	LSB_OLD_JOBID
LSB_OUTPUT_TARGETFAILED	LSB_QUEUE	LSB_REMOTEINDEX
LSB_REMOTEJID	LSB_RESTART	LSB_RESTART_PGID
LSB_RESTART_PID	LSB_SUB_CLUSTER	LSB_SUB_COMMAND_LINE
LSB_SUB_EXTSCHED_PARAM	LSB_SUB_JOB_WARNING_ACTION	LSB_SUB_JOB_ACTION_WARNING_TIME
LSB_SUB_PARM_FILE	LSB_SUSP_REASONS	LSB_SUSP_SUBREASONS
LSF_CMD_LOGDIR	LSF_DEBUG_CMD	LSF_DEBUG_LIM
LSF_DEBUG_RES	LSF_EAUTH_AUX_DATA	LSF_EAUTH_AUX_PASS
LSF_EAUTH_CLIENT	LSF_EAUTH_SERVER	LSF_EAUTH_UID
LSF_INTERACTIVE_STDERR	LSF_INVOKE_CMD	LSF_JOB_STARTER
LSF_LIM_DEBUG	LSF_LOGDIR	LSF_MASTER
LSF_NIOS_DEBUG	LSF_NIOS_DIE_CMD	LSF_NIOS_IGNORE_SIGWINDOW
LSF_NIOS_PEND_TIMEOUT	LSF_RESOURCES	LSF_USE_HOSTEQUIV
LSF_USER_DOMAIN		

BSUB_BLOCK

Description If set, tells NIOS that it is running in batch mode.

Default Undefined

Notes If you submit a job with the `-K` option of `bsub`, which is synchronous execution, then BSUB_BLOCK is set. Synchronous execution means you have to wait for the job to finish before you can continue.

Where defined Set internally

See also The `-K` option of `bsub`

BSUB_QUIET

Syntax **`BSUB_QUIET=`***`any_value`*

Description Controls the printing of information about job submissions. If set, `bsub` will not print any information about job submission. For example, it will not print `<Job is submitted to default queue <normal>`, nor `<Waiting for dispatch>`.

Default Undefined

Where defined From the command line

Example `BSUB_QUIET=1`

BSUB_QUIET2

Syntax **BSUB_QUIET2=***any_value*

Description Suppresses the printing of information about job completion when a job is submitted with the `bsub -K` option.

If set, `bsub` will not print information about job completion to `stdout`. For example, when this variable is set, the message <<Job is finished>> will not be written to `stdout`.

If BSUB_QUIET and BSUB_QUIET2 are both set, no job messages will be printed to `stdout`.

Default Undefined

Where defined From the command line

Example `BSUB_QUIET2=1`

BSUB_STDERR

Syntax **BSUB_STDERR=y**

Description Redirects LSF messages for `bsub` to `stderr`.

By default, when this parameter is not set, LSF messages for `bsub` are printed to `stdout.`

When this parameter is set, LSF messages for `bsub` are redirected to `stderr`.

Default Undefined

Where defined From the command line on UNIX. For example, in `csh`:

```
setenv BSUB_STDERR Y
```

From the Control Panel on Windows, as an environment variable

CLEARCASE_DRIVE

Syntax **CLEARCASE_DRIVE=***drive_letter***:**

Description Optional, Windows only.

Defines the virtual drive letter for a Rational ClearCase view to the drive. This is useful if you wish to map a Rational ClearCase view to a virtual drive as an alias.

If this letter is unavailable, Windows attempts to map to another drive. Therefore, CLEARCASE_DRIVE only defines the default drive letter to which the Rational ClearCase view is mapped, not the final selected drive letter. However, the PATH value is automatically updated to the final drive letter if it is different from CLEARCASE_DRIVE.

Notes: CLEARCASE_DRIVE is case insensitive.

Where defined From the command line

Example `CLEARCASE_DRIVE=F:`

`CLEARCASE_DRIVE=f:`

See also CLEARCASE_MOUNTDIR, CLEARCASE_ROOT.

CLEARCASE_MOUNTDIR

Syntax **`CLEARCASE_MOUNTDIR=`***`path`*

Description Optional.

Defines the Rational ClearCase mounting directory.

Default `/vobs`

Notes: CLEARCASE_MOUNTDIR is used if any of the following conditions apply:

- A job is submitted from a UNIX environment but run in a Windows host.
- The Rational ClearCase mounting directory is not the default `/vobs`

Where defined From the command line

Example `CLEARCASE_MOUNTDIR=/myvobs`

See also CLEARCASE_DRIVE, CLEARCASE_ROOT.

CLEARCASE_ROOT

Syntax **`CLEARCASE_ROOT=`***`path`*

Description The path to the Rational ClearCase view.

In Windows, this path must define an absolute path starting with the default ClearCase drive and ending with the view name without an ending backslash (\).

Notes CLEARCASE_ROOT must be defined if you want to submit a batch job from a ClearCase view.

For interactive jobs, use bsub -I to submit the job.

Where defined In the job starter, or from the command line

Example In UNIX:

`CLEARCASE_ROOT=/view/myview`

In Windows:

`CLEARCASE_ROOT=F:\myview`

See also CLEARCASE_DRIVE, CLEARCASE_MOUNTDIR, LSF_JOB_STARTER

LM_LICENSE_FILE

Syntax **`LM_LICENSE_FILE=`***`file_name`*

Description The path to where the license file is found. The file name is the name of the license file.

Default `/usr/share/flexlm/licenses/license.dat`

Notes A FLEXnet variable read by the `lmgrd` daemon.

Where defined From the command line

See Also See "lsf.conf" under "LSF_LICENSE_FILE" on page 553

LS_EXEC_T

Syntax **LS_EXEC_T= START | END | CHKPNT | JOB_CONTROLS**

Description Indicates execution type for a job. LS_EXEC_T is set to:

- START or END for a job when the job begins executing or when it completes execution
- CHKPNT when the job is checkpointed
- JOB_CONTROLS when a control action is initiated

Where defined Set by `sbatchd` during job execution

LS_JOBPID

Description The process ID of the job.

Where defined During job execution, `sbatchd` sets LS_JOBPID to be the same as the process ID assigned by the operating system.

LS_LICENSE_SERVER_*feature*

Syntax **LS_LICENSE_SERVER_*feature*="***domain***:***server***:***num_available* ...**"**

server is of the format *port***@***host*

Description The license server information provided to the job. The purpose of this environment variable is to provide license server information to the job.

Where defined During the license job execution, `sbatchd` sets LS_LICENSE_SERVER_*feature* to be the same as the license server information defined in the job's `rusage` string. This is only used by the job and logged in the `mbatchd` log file if DEBUG1 and LC_LICSCHED are defined in `lsf.conf`.

LS_SUBCWD

Description The current working directory (`cwd`) of the submission host where the remote task command was executed.

The way this parameter is set by LSF is as follows:

1. LSF looks for the PWD environment variable. If it finds it, sets LS_SUBCWD to PWD.
2. If the PWD environment variable does not exist, LSF looks for the CWD environment variable. If it finds CWD, sets LS_SUBCWD to CWD.
3. If the CWD environment variable does not exist, LSF calls the `getwd()` system function to retrieve the current working directory path name. LSF sets LS_SUBCWD to the value that is returned.

Where defined Set by `sbatchd`

LSB_CHKPNT_DIR

Syntax **LSB_CHKPNT_DIR=***checkpoint_dir/job_ID*

Description The directory containing files related to the submitted checkpointable job.

Valid values The value of `checkpoint_dir` is the directory you specified through the -k option of `bsub` when submitting the checkpointable job.

The value of `job_ID` is the job ID of the checkpointable job.

Where defined Set by LSF, based on the directory you specified when submitting a checkpointable job with the `-k` option of `bsub`.

LSB_DEBUG

This parameter can be set from the command line or from `lsf.conf`. See "lsf.conf" under "LSB_DEBUG" on page 512.

LSB_DEBUG_CMD

This parameter can be set from the command line or from `lsf.conf`. See "lsf.conf" under "LSB_DEBUG_CMD" on page 512.

LSB_DEBUG_MBD

This parameter can be set from the command line with `badmin mbddebug` or from `lsf.conf`.

See "lsf.conf" under "LSB_DEBUG_MBD" on page 513.

LSB_DEBUG_NQS

This parameter can be set from the command line or from `lsf.conf`. See "lsf.conf" under "LSB_DEBUG_NQS" on page 514.

LSB_DEBUG_SBD

This parameter can be set from the command line with `badmin sbddebug` or from `lsf.conf`.

See "lsf.conf" under "LSB_DEBUG_SBD" on page 514.

LSB_DEBUG_SCH

This parameter can be set from the command line or from `lsf.conf`. See "lsf.conf" under "LSB_DEBUG_SCH" on page 515.

LSB_DEFAULTPROJECT

Syntax **`LSB_DEFAULTPROJECT=`**_`project_name`_

Description The name of the project to which resources consumed by a job will be charged.

Default Undefined

Notes If the LSF administrator defines a default project in the `lsb.params` configuration file, the system uses this as the default project. You can change the default project by setting LSB_DEFAULTPROJECT or by specifying a project name with the `-P` option of `bsub`.

If you submit a job without the `-P` option of `bsub`, but you defined LSB_DEFAULTPROJECT, then the job belongs to the project specified in LSB_DEFAULTPROJECT.

If you submit a job with the `-P` option of `bsub`, the job belongs to the project specified through the `-P` option.

Where defined From the command line, or through the `-P` option of `bsub`

Example `LSB_DEFAULTPROJECT=engineering`

See also See "lsb.params" under "DEFAULT_PROJECT" on page 378, the `-P` option of `bsub`.

LSB_DEFAULTQUEUE

Syntax **`LSB_DEFAULTQUEUE=`***`queue_name`*

Description Defines the default LSF queue.

Default `mbatchd` decides which is the default queue. You can override the default by defining LSB_DEFAULTQUEUE.

Notes If the LSF administrator defines a default queue in the `lsb.params` configuration file, then the system uses this as the default queue. Provided you have permission, you can change the default queue by setting LSB_DEFAULTQUEUE to a valid queue (see `bqueues` for a list of valid queues).

Where defined From the command line

See also See "lsb.params" under "DEFAULT_QUEUE" on page 378.

LSB_ECHKPNT_METHOD

This parameter can be set as an environment variable and/or in `lsf.conf`. See "lsf.conf" under "LSB_ECHKPNT_METHOD" on page 517.

LSB_ECHKPNT_METHOD_DIR

This parameter can be set as an environment variable and/or in `lsf.conf`. See "lsf.conf" under "LSB_ECHKPNT_METHOD_DIR" on page 518.

LSB_ECHKPNT_KEEP_OUTPUT

This parameter can be set as an environment variable and/or in `lsf.conf`. See "lsf.conf" under "LSB_ECHKPNT_KEEP_OUTPUT" on page 517.

LSB_ERESTART_USRCMD

Syntax **`LSB_ERESTART_USRCMD=`***`command`*

Description Original command used to start the job.

This environment variable is set by `erestart` to pass the job's original start command to a custom `erestart` method `erestart.`*`method_name`*. The value of this variable is extracted from the job file of the checkpointed job.

If a job starter is defined for the queue to which the job was submitted, the job starter is also included in LSB_ERESTART_USRCMD. For example, if the job starter is `/bin/sh -c "%USRCMD"` in `lsb.queues`, and the job name is `myapp -d`, LSB_ERESTART_USRCMD will be set to:

```
/bin/sh -c "myapp -d"
```

Where defined Set by `erestart` as an environment variable before a job is restarted

See also LSB_ECHKPNT_METHOD, `erestart`, `echkpnt`

LSB_EXEC_RUSAGE

Syntax **`LSB_EXEC_RUSAGE="`***`resource_name1 resource_value1 resource_name2 resource_value2...`***`"`**

Description Indicates which `rusage` string is satisfied to permit the job to run. This environment variable is necessary because the `OR` (||) operator specifies alternative `rusage` strings for running jobs.

Valid values `resource_value1`, `resource_value2`,... refer to the resource values on `resource_name1`, `resource_name2`,... respectively.

Default Undefined

Where defined Set by LSF after reserving a resource for the job.

LSB_EXECHOSTS

Description A list of hosts on which a batch job will run.

Where defined Set by `sbatchd`

Product MultiCluster

LSB_EXIT_PRE_ABORT

Description The queue-level or job-level `pre_exec_command` can exit with this value if the job is to be aborted instead of being requeued or executed

Where defined Set by `sbatchd`

See also See "lsb.queues", or the `-E` option of `bsub`.

LSB_EXIT_REQUEUE

Syntax **`LSB_EXIT_REQUEUE="`***`exit_value1 exit_value2...`***`"`**

Description Contains a list of exit values found in the queue's REQUEUE_EXIT_VALUES parameter defined in `lsb.queues`.

Valid Values Any positive integers

Default Undefined

Notes If LSB_EXIT_REQUEUE is defined, a job will be requeued if it exits with one of the specified values.

LSB_EXIT_REQUEUE is undefined if the parameter REQUEUE_EXIT_VALUES is undefined.

Where defined Set by the system based on the value of the parameter REQUEUE_EXIT_VALUES in `lsb.queues`

Example `LSB_EXIT_REQUEUE="7 31"`

See also See "lsb.queues" under "REQUEUE_EXIT_VALUES" on page 421.

LSB_FRAMES

Syntax **LSB_FRAMES=***start_number***,***end_number***,***step*

Description Determines the number of frames to be processed by a frame job.

Valid values The values of *start_number*, *end_number*, and *step* are positive integers. Use commas to separate the values.

Default Undefined

Notes When the job is running, LSB_FRAMES will be set to the relative frames with the format LSB_FRAMES=*start_number*,*end_number*,*step*.

From the *start_number*, *end_number*, and *step*, the frame job can know how many frames it will process.

Where defined Set by `sbatchd`

Example `LSB_FRAMES=10,20,1`

LSB_HOSTS

Syntax **LSB_HOSTS="***host_name...***"**

Description A list of hosts selected by LSF Batch to run the batch job.

Notes If a job is run on a single processor, the system sets LSB_HOSTS to the name of the host used.

For parallel jobs, the system sets LSB_HOSTS to the names of all the hosts used.

Where defined Set by `sbatchd` when the job is executed. LSB_HOSTS is set only when the list of host names is less than 4096 bytes.

See also See LSB_MCPU_HOSTS.

LSB_INTERACTIVE

Syntax **LSB_INTERACTIVE=Y**

Description Indicates an interactive job. When you submit an interactive job using `bsub -I`, the system sets LSB_INTERACTIVE to Y.

Valid values LSB_INTERACTIVE=Y (if the job is interactive)

Default Undefined (if the job is not interactive)

Where defined Set by `sbatchd`

LSB_JOB_STARTER

Syntax **LSB_JOB_STARTER=***binary*

Description Specifies an executable program that has the actual job as an argument.

Default Undefined

Notes
- Interactive Jobs

If you want to run an interactive job that requires some preliminary setup, LSF provides a job starter function at the command level. A command-level job starter allows you to specify an executable file that will run prior to the actual job, doing any necessary setup and running the job when the setup is complete.

If the environment variable LSB_JOB_STARTER is properly defined, `sbatchd` will invoke the job starter (rather than the job itself), supplying your commands as arguments.

- Batch Jobs

  A job starter can also be defined at the queue level using the JOB_STARTER parameter, although this can only be done by the LSF administrator.

Where defined From the command line

See also See "lsb.queues" under "JOB_STARTER" on page 412.

Example

- UNIX

  The job starter is invoked from within a Bourne shell, making the command-line equivalent:

  `/bin/sh -c "$LSB_JOB_STARTER` *`command`* `[`*`argument...`*`]"`

  where `command [argument...]` are the command line arguments you specified in `lsrun`, `lsgrun`, or `ch`.

  If you define LSB_JOB_STARTER as follows:

  ```
  % setenv LSB_JOB_STARTER "/bin/csh -c"
  ```

  and run a simple C-shell job:

  ```
  % lsrun "'a.out; echo hi'"
  ```

  then the following will be invoked to correctly start the job:

  ```
  /bin/sh -c "/bin/csh -c 'a.out; echo hi'"
  ```

- Windows

  RES runs the job starter, passing it your commands as arguments:

  `LSB_JOB_STARTER` *`command`* `[`*`argument...`*`]`

  If you define LSB_JOB_STARTER as follows:

  ```
  set LSB_JOB_STARTER=C:\cmd.exe /C
  ```

  and run a simple DOS shell job:

  ```
  C:\> lsrun dir /p
  ```

  then the following will be invoked to correctly start the job:

  ```
  C:\cmd.exe /C dir /p
  ```

See also See "lsb.queues" under "JOB_STARTER" on page 412.

LSB_JOBEXIT_INFO

Syntax **`LSB_JOBEXIT_INFO="SIGNAL`** *`signal_value signal_name`***`"`**

Description Contains information about signal that caused a job to exit.

Applies to post-execution commands. Post-execution commands are set with POST_EXEC in `lsb.queues`.

When the post-execution command is run, the environment variable LSB_JOBEXIT_INFO is set if the job is signalled internally. If the job ends successfully, or the job is killed or signalled externally, LSB_JOBEXIT_INFO is not set.

Examples

```
LSB_JOBEXIT_INFO="SIGNAL -1 SIG_CHKPNT"
LSB_JOBEXIT_INFO="SIGNAL -14 SIG_TERM_USER"
LSB_JOBEXIT_INFO="SIGNAL -23 SIG_KILL_REQUEUE"
```

Default Undefined

Where defined Set by `sbatchd`

LSB_JOBEXIT_STAT

Syntax **`LSB_JOBEXIT_STAT=`***`exit_status`*

Description Indicates a job's exit status.

Applies to post-execution commands. Post-execution commands are set with POST_EXEC in `lsb.queues`.

When the post-execution command is run, the environment variable LSB_JOBEXIT_STAT is set to the exit status of the job. Refer to the man page for the `wait`(2) command for the format of this exit status.

The post-execution command is also run if a job is requeued because the job's execution environment fails to be set up, or if the job exits with one of the queue's REQUEUE_EXIT_VALUES. The LSB_JOBPEND environment variable is set if the job is requeued. If the job's execution environment could not be set up, LSB_JOBEXIT_STAT is set to 0.

Valid values Any positive integer

Where defined Set by `sbatchd`

LSB_JOBFILENAME

Syntax **`LSB_JOBFILENAME=`***`file_name`*

Description The path to the batch job file.

Notes Specifies the path to the batch executable job file that invokes the batch job. The batch executable job file is a `/bin/sh` script on UNIX systems or a `.BAT` command script on Windows systems.

LSB_JOBID

Syntax **`LSB_JOBID=`***`job_ID`*

Description The job ID assigned by the Batch system. This is the ID of the job assigned by LSF, as shown by `bjobs`.

Valid values Any positive integer

Where defined Set by `sbatchd`, defined by `mbatchd`

See also LSB_REMOTEJID

LSB_JOBINDEX

Syntax **LSB_JOBINDEX=***index*

Description Contains the job array index.

Valid values Any integer greater than zero but less than the maximum job array size.

Notes LSB_JOBINDEX is set when each job array element is dispatched. Its value corresponds to the job array index. LSB_JOBINDEX is set for all jobs. For non-array jobs, LSB_JOBINDEX is set to zero (0).

Where defined Set during job execution based on `bsub` options.

Example You can use LSB_JOBINDEX in a shell script to select the job command to be performed based on the job array index.

For example:

```
if [$LSB_JOBINDEX -eq 1]; then
cmd1
fi
if [$LSB_JOBINDEX -eq 2]; then
cmd2
fi
```

See also LSB_JOBINDEX_STEP, LSB_REMOTEINDEX

LSB_JOBINDEX_STEP

Syntax **LSB_JOBINDEX_STEP=***step*

Description Step at which single elements of the job array are defined.

Valid values Any integer greater than zero but less than the maximum job array size

Default 1

Notes LSB_JOBINDEX_STEP is set when a job array is dispatched. Its value corresponds to the step of the job array index. This variable is set only for job arrays.

Where defined Set during job execution based on `bsub` options.

Example The following is an example of an array where a step of 2 is used:

```
array[1-10:2]
elements:1 3 5 7 9
```

If this job array is dispatched, then LSB_JOBINDEX_STEP=2

See also LSB_JOBINDEX

LSB_JOBNAME

Syntax **LSB_JOBNAME=***job_name*

Description The name of the job defined by the user at submission time.

Default The job’s command line

Notes The name of a job can be specified explicitly when you submit a job. The name does not have to be unique. If you do not specify a job name, the job name defaults to the actual batch command as specified on the `bsub` command line.

Where defined Set by `sbatchd`

Example When you submit a job using the `-J` option of `bsub`, for example:

```
% bsub -J "myjob" job
```

`sbatchd` sets LSB_JOBNAME to the job name that you specified:

```
LSB_JOBNAME=myjob
```

LSB_JOBPEND

Description Set if the job is requeued.

Where defined Set by `sbatchd` for POST_EXEC only

See also LSB_JOBEXIT_STAT, REQUEUE_EXIT_VALUES, POST_EXEC.

LSB_JOBPGIDS

Description A list of the current process group IDs of the job.

Where defined The process group IDs are assigned by the operating system, and LSB_JOBPGIDS is set by `sbatchd`.

See also LSB_JOBPIDS

LSB_JOBPIDS

Description A list of the current process IDs of the job.

Where defined The process IDs are assigned by the operating system, and LSB_JOBPIDS is set by `sbatchd`.

See also LSB_JOBPGIDS

LSB_MAILSIZE

Syntax **`LSB_MAILSIZE=`***`value`*

Description Gives an estimate of the size of the batch job output when the output is sent by email. It is not necessary to configure LSB_MAILSIZE_LIMIT.

LSF sets LSB_MAILSIZE to the size in KB of the job output, allowing the custom mail program to intercept output that is larger than desired.

LSB_MAILSIZE is not recognized by the LSF default mail program. To prevent large job output files from interfering with your mail system, use LSB_MAILSIZE_LIMIT to explicitly set the maximum size in KB of the email containing the job information.

Valid values

- A positive integer

  If the output is being sent by email, LSB_MAILSIZE is set to the estimated mail size in kilobytes.

- -1

If the output fails or cannot be read, LSB_MAILSIZE is set to -1 and the output is sent by email using LSB_MAILPROG if specified in `lsf.conf`.

- Undefined

  If you use the `-o` or `-e` options of `bsub`, the output is redirected to an output file. Because the output is not sent by email in this case, LSB_MAILSIZE is not used and LSB_MAILPROG is not called.

  If the `-N` option is used with the `-o` option of `bsub`, LSB_MAILSIZE is not set.

Where defined Set by `sbatchd` when the custom mail program specified by LSB_MAILPROG in `lsf.conf` is called.

LSB_MCPU_HOSTS

Syntax **`LSB_MCPU_HOSTS="`**`host_nameA num_processors1 host_nameB num_processors2...`**`"`**

Description Contains a list of the hosts and the number of CPUs used to run a job.

Valid values `num_processors1`, `num_processors2`,... refer to the number of CPUs used on `host_nameA`, `host_nameB`,..., respectively

Default Undefined

Notes The environment variables LSB_HOSTS and LSB_MCPU_HOSTS both contain the same information, but the information is presented in different formats. LSB_MCPU_HOSTS uses a shorter format than LSB_HOSTS. As a general rule, `sbatchd` sets both these variables. However, for some parallel jobs, LSB_HOSTS is not set.

For parallel jobs, several CPUs are used, and the length of LSB_HOSTS can become very long. `sbatchd` needs to spend a lot of time parsing the string. If the size of LSB_HOSTS exceeds 4096 bytes, LSB_HOSTS is ignored, and `sbatchd` sets only LSB_MCPU_HOSTS.

To verify the hosts and CPUs used for your dispatched job, check the value of LSB_HOSTS for single CPU jobs, and check the value of LSB_MCPU_HOSTS for parallel jobs.

Where defined Set by `sbatchd` before starting a job on the execution host

Example When the you submit a job with the `-m` and `-n` options of `bsub`, for example,

```
% bsub -m "hostA hostB" -n 6 job
```

`sbatchd` sets the environment variables LSB_HOSTS and LSB_MCPU_HOSTS as follows:

```
LSB_HOSTS= "hostA hostA hostA hostB hostB hostB"
```

```
LSB_MCPU_HOSTS="hostA 3 hostB 3"
```

Both variables are set in order to maintain compatibility with earlier versions.

See also LSB_HOSTS

LSB_NQS_PORT

This parameter can be defined in lsf.conf or in the services database such as /etc/services.

See "lsf.conf" under "LSB_NQS_PORT" on page 528 for more details.

LSB_NTRIES

Syntax **LSB_NTRIES=***integer*

Description The number of times that LSF libraries attempt to contact mbatchd or perform a concurrent jobs query.

For example, if this parameter is undefined, when you type bjobs, LSF keeps displaying "batch system not responding" if mbatchd cannot be contacted or if the number of pending jobs exceeds MAX_PEND_JOBS specified in lsb.params or lsb.users.

If this parameter is set to a value, LSF only attempts to contact mbatchd the defined number of times and then quits. LSF will wait for a period of time equal to SUB_TRY_INTERVAL specified in lsb.params before attempting to contact mbatchd again.

Valid values Any positive integer

Default INFINIT_INT (The default is to continue the attempts to contact mbatchd)

LSB_OLD_JOBID

Syntax **LSB_OLD_JOBID=***job_ID*

Description The job ID of a job at the time it was checkpointed.

When a job is restarted, it is assigned a new job ID and LSB_JOBID is replaced with the new job ID. LSB_OLD_JOBID identifies the original ID of a job before it is restarted.

Valid values Any positive integer

Where defined Set by sbatchd, defined by mbatchd

See also LSB_JOBID

LSB_OUTPUT_TARGETFAILED

Syntax **LSB_OUTPUT_TARGETFAILED=Y**

Description Indicates that LSF cannot access the output file specified for a job submitted the bsub -o option.

Valid values Set to Y if the output file cannot be accessed; otherwise, it is undefined.

Where defined Set by sbatchd during job execution

LSB_QUEUE

Description The name of the queue from which the job is dispatched.

Where defined Set by sbatchd

LSB_REMOTEINDEX

Syntax **LSB_REMOTEINDEX=***index*

Description The job array index of a remote MultiCluster job. LSB_REMOTEINDEX is set only if the job is an element of a job array.

Valid values Any integer greater than zero, but less than the maximum job array size

Where defined Set by `sbatchd`

See also LSB_JOBINDEX, "MAX_JOB_ARRAY_SIZE" on page 385 in "lsb.params"

LSB_REMOTEJID

Syntax **LSB_REMOTEJID=***job_ID*

Description The job ID of a remote MultiCluster job.

Where defined Set by `sbatchd`, defined by `mbatchd`

See also LSB_JOBID

LSB_RESTART

Syntax **LSB_RESTART=Y**

Description Indicates that a job has been restarted or migrated.

Valid values Set to Y if the job has been restarted or migrated; otherwise, it is undefined.

Notes If a batch job is submitted with the `-r` option of `bsub`, and is restarted because of host failure, then LSB_RESTART is set to Y. If a checkpointable job is submitted with the `-k` option of `bsub`, then LSB_RESTART is set to Y when the job is restarted. If `bmig` is used to migrate a job, then LSB_RESTART is set to Y when the migrated job is restarted.

If the job is not a restarted job, then LSB_RESTART is not set.

Where defined Set by `sbatchd` during job execution

See also LSB_RESTART_PGID, LSB_RESTART_PID

LSB_RESTART_PGID

Syntax **LSB_RESTART_PGID=***pgid*

Description The process group ID of the checkpointed job when the job is restarted.

Notes When a checkpointed job is restarted, the operating system assigns a new group process ID to the job. Batch sets LSB_RESTART_PGID to the new group process ID.

Where defined Set by Batch during restart of a checkpointed job.

See also LSB_RESTART_PID, LSB_RESTART

LSB_RESTART_PID

Syntax **LSB_RESTART_PID=***pid*

Description The process ID of the checkpointed job when the job is restarted.

Notes When a checkpointed job is restarted, the operating system assigns a new process ID to the job. Batch sets LSB_RESTART_PID to the new process ID.

Where defined Defined by Batch during restart of a checkpointed job

See also LSB_RESTART_PGID, LSB_RESTART

LSB_SUB_CLUSTER

Description Name of submission cluster (MultiCluster only)

Where defined Set on the submission environment and passed to the execution cluster environment. The parameter will ONLY be valid in Multi Cluster environment. For jobs on a local cluster, the parameter is not set when using any daemon wrappers such as job starter, post-, pre- or eexec scripts.

LSB_SUB_COMMAND_LINE

Description The job command line.

Where defined Set by `esub` before a job is submitted.

LSB_SUB_EXTSCHED_PARAM

Description Value of external scheduling options specified by `bsub -extsched`, or queue-level MANDATORY_EXTSCHED or DEFAULT_EXTSCHED.

Where defined Set by `esub` before a job is submitted.

LSB_SUB_JOB_WARNING_ACTION

Description Value of job warning action specified by `bsub -wa`.

Where defined Set by `esub` before a job is submitted.

LSB_SUB_JOB_ACTION_WARNING_TIME

Description Value of job warning time period specified by `bsub -wt`.

Where defined Set by `esub` before a job is submitted.

LSB_SUB_PARM_FILE

Usage LSB_SUB_PARM_FILE=*file_name*

Description Indicates to `esub` the file in which the job submission parameters are written

Notes Points to a file in which the job submission parameters are written. The submission parameters are a set of name-value pairs on separate lines in the format `"`*`option_name=value`*`"`. A typical use of this file is to control job submission options.

Where defined Set by LSF on the submission host before running `esub`. Not defined when `esub` is invoked in interactive remote execution.

LSB_SUSP_REASONS

Syntax **`LSB_SUSP_REASONS=`***`integer`*

Description An integer representing suspend reasons. Suspend reasons are defined in `lsbatch.h`.

This parameter is set when a job goes to system-suspended (SSUSP) or user-suspended status (USUSP). It indicates the exact reason why the job was suspended.

To determine the exact reason, you can test the value of LSB_SUSP_REASONS against the symbols defined in `lsbatch.h`.

Default Undefined

Where defined Set by `sbatchd`

See Also LSB_SUSP_SUBREASONS

LSB_SUSP_SUBREASONS

Syntax **`LSB_SUSP_SUBREASONS=`***`integer`*

Description An integer representing the load index that caused a job to be suspended.

When the suspending reason SUSP_LOAD_REASON (suspended by load) is set in LSB_SUSP_REASONS, LSB_SUSP_SUBREASONS set to one of the load index values defined in `lsf.h`.

Use LSB_SUSP_REASONS and LSB_SUSP_SUBREASONS together in you custom job control to determine the exact load threshold that caused a job to be suspended.

Load index values are defined in `lsf.h`.

Load Index	Value
R15S	0
R1M	1
R15M	2
UT	3
PG	4
IO	5
LS	6
IT	7
TMP	8
SWP	9
MEM	10

Default Undefined

Where defined Set by `sbatchd`

See also LSB_SUSP_REASONS

LSF_CMD_LOGDIR

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_CMD_LOGDIR" on page 538.

LSF_DEBUG_CMD

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSB_DEBUG_MBD" on page 513.

LSF_DEBUG_LIM

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_DEBUG_LIM" on page 539.

LSF_DEBUG_RES

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_DEBUG_RES" on page 540.

LSF_EAUTH_AUX_DATA

Syntax **`LSF_EAUTH_AUX_DATA=`***`path/file_name`*

Description The full path to the temporary file on the local file system that is used for storing auxiliary authentication information.

Notes Credentials are passed between invocations of `eauth` and the daemons through the file defined by LSF_EAUTH_AUX_DATA.

To allow daemons to call `eauth` to authenticate each other, you must define LSF_AUTH_DAEMONS.

Where defined Set internally by `eauth`

See also LSF_AUTH_DAEMONS, LSF_EAUTH_AUX_PASS

LSF_EAUTH_AUX_PASS

Syntax **`LSF_EAUTH_AUX_PASS=yes`**

Description Grants permission.

LSF_EAUTH_AUX_PASS is passed to `eauth -c` when LSF_EAUTH_CLIENT=user, and it tells `eauth` that it has permission to forward auxiliary authentication data.

To allow daemons to call `eauth` to authenticate each other, you must define LSF_AUTH_DAEMONS.

Where defined Set internally

Product SUN HPC

See also LSF_EAUTH_AUX_DATA, LSF_EAUTH_CLIENT

LSF_EAUTH_CLIENT

Syntax

- SUN HPC

  `LSF_EAUTH_CLIENT=mbatchd | sbatchd | pam | res | user`

- LSF3.2+

  `LSF_EAUTH_CLIENT=user`

Description A string that specifies the daemon or user that is calling `eauth -c`.

Notes Sets the context for the call to `eauth`, and allows the `eauth` writer to perform daemon authentication.

Where defined Set internally by the LSF libraries, or by the daemon calling `eauth -c`.

See also LSF_EAUTH_SERVER

LSF_EAUTH_SERVER

Syntax
- SUN HPC

  `LSF_EAUTH_SERVER=mbatchd | sbatchd | pam | res`
- LSF3.2+

  `LSF_EAUTH_SERVER=mbatchd | res`

Description Specifies the daemon or user that is calling `eauth -s`

Notes Sets the context for the call to `eauth`, and allows the `eauth` writer to perform daemon authentication.

Where defined Set internally by the LSF libraries, or by the daemon calling `eauth -s`

See also LSF_EAUTH_CLIENT

LSF_EAUTH_UID

Syntax **`LSF_EAUTH_UID=`***`user_ID`*

Description Specifies the user ID under which `eauth -s` must run.

Where defined Set by the LSF daemon which executes `eauth`.

See also See "LSF_EAUTH_USER" on page 613 in "lsf.sudoers".

LSF_INTERACTIVE_STDERR

This parameter can be defined in `lsf.conf`.

See "lsf.conf" under "LSF_INTERACTIVE_STDERR" on page 550 for more details.

LSF_INVOKE_CMD

Usage `LSF_INVOKE_CMD=`*`invoking_command_name`*

Description Indicates the name of the last LSF command that invoked an external executable (for example, `esub`).

External executables get called by several LSF commands (`bsub`, `bmod`, `lsrun`). This variable contians the name of the last LSF command to call the executable.

Default Undefined

Where defined Set internally within the LSF library.

LSF_JOB_STARTER

Syntax **`LSF_JOB_STARTER=`***`binary`*

Description Specifies an executable program that has the actual job as an argument.

Default Undefined

Notes Interactive Jobs

If you want to run an interactive job that requires some preliminary setup, LSF provides a job starter function at the command level. A command-level job starter allows you to specify an executable file that will run prior to the actual job, doing any necessary setup and running the job when the setup is complete.

If LSF_JOB_STARTER is properly defined, RES will invoke the job starter (rather than the job itself), supplying your commands as arguments.

Batch Jobs

A job starter can also be defined at the queue level using the JOB_STARTER parameter, although this can only be done by the LSF administrator.

Where defined From the command line

Example

- UNIX

  The job starter is invoked from within a Bourne shell, making the command-line equivalent:

  `/bin/sh -c "$LSF_JOB_STARTER` *`command`* `[`*`argument...`*`]"`

  where `command [argument...]` are the command line arguments you specified in `lsrun`, `lsgrun`, or `ch`.

  If you define LSF_JOB_STARTER as follows:

  ```
  % setenv LSF_JOB_STARTER "/bin/csh -c"
  ```

  and run a simple C-shell job:

  ```
  % lsrun "'a.out; echo hi'"
  ```

  The following will be invoked to correctly start the job:

  ```
  /bin/sh -c "/bin/csh -c 'a.out; echo hi'"
  ```

- Windows

  RES runs the job starter, passing it your commands as arguments:

  ```
  LSF_JOB_STARTER command [argument...]
  ```

  If you define LSF_JOB_STARTER as follows:

  ```
  set LSF_JOB_STARTER=C:\cmd.exe /C
  ```

  and run a simple DOS shell job:

  ```
  C:\> lsrun dir /p
  ```

  then the following will be invoked to correctly start the job:

  ```
  C:\cmd.exe /C dir /p
  ```

See also See "lsb.queues" under "JOB_STARTER" on page 412.

LSF_LIM_DEBUG

This parameter can be set from the command line or from `lsf.conf`.

See "lsf.conf" under "LSF_LIM_DEBUG" on page 554.

LSF_LOGDIR

This parameter can be set from the command line or from `lsf.conf`.

See “lsf.conf” under “LSF_LOGDIR” on page 558.

LSF_MASTER

Description Specifies whether ELIM has been started on the master host.

Notes LIM communicates with ELIM through two environment variables: LSF_MASTER and LSF_RESOURCES.

LSF_MASTER is set to `Y` when LIM starts ELIM on the master host. It is set to N or is undefined otherwise.

LSF_MASTER can be used to test whether the ELIM should report on cluster-wide resources that only need to be collected on the master host.

When defined Set by LIM when ELIM is started

See also LSF_RESOURCES

LSF_NIOS_DEBUG

This parameter can be set from the command line or from `lsf.conf`. See “lsf.conf” under “LSF_NIOS_DEBUG” on page 561.

LSF_NIOS_DIE_CMD

Syntax **`LSF_NIOS_DIE_CMD=`***`command`*

Description If set, the command defined by LSF_NIOS_DIE_CMD is executed before NIOS exits.

Default Undefined

Where defined From the command line

LSF_NIOS_IGNORE_SIGWINDOW

Syntax **`LSF_NIOS_IGNORE_SIGWINDOW=`***`any_value`*

Description If defined, the NIOS will ignore the SIGWINDOW signal.

Default Undefined

Notes When the signal SIGWINDOW is defined, some tasks appear to die when they receive the SIGWINDOW while doing I/O. By defining LSF_NIOS_IGNORE_SIGWINDOW, these tasks are given the chance to ignore the signal.

Where defined From the command line

LSF_NIOS_PEND_TIMEOUT

Syntax **`LSF_NIOS_PEND_TIMEOUT=`***minutes*

Description Applies only to interactive batch jobs.

Maximum amount of time that an interactive batch job can remain pending.

If this parameter is defined, and an interactive batch job is pending for longer than the specified time, the interactive batch job is terminated.

Valid values Any integer greater than zero

Default Undefined

LSF_RESOURCES

Syntax **LSF_RESOURCES=***dynamic_shared_resource_name...*

Description Space-separated list of customized dynamic shared resources that the ELIM is responsible for collecting.

Valid values A resource name is only put in the list if the host on which the ELIM is running shares an instance of that resource.

Notes LIM communicates with the ELIM through two environment variables: LSF_MASTER and LSF_RESOURCES.

LSF_MASTER is set to `Y` when LIM starts ELIM on the master host. It is set to N or is undefined otherwise.

LSF_RESOURCES is set to a space-separated string of dynamic shared resources for which the ELIM on that host is responsible for collecting. LSF_RESOURCES gets passed to ELIM from LIM.

When defined By LIM when ELIM is invoked

Example `LSF_RESOURCES="resource1 resource2 resource3"`

See also LSF_MASTER

LSF_USE_HOSTEQUIV

Syntax **LSF_USE_HOSTEQUIV=y** | **Y**

Description Used for authentication purposes. If LSF_USE_HOSTEQUIV is defined, RES and `mbatchd` call the `ruserok`(3) function to decide if a user is allowed to run remote jobs. LSF trusts all hosts configured in the LSF cluster that are defined in `hosts.equiv`, or in `.rhosts` in the user's home directory.

The `ruserok`(3) function checks in the `/etc/hosts.equiv` file and the user's `$HOME/.rhosts` file to decide if the user has permission to execute remote jobs.

If LSF_USE_HOSTEQUIV is not defined, all normal users in the cluster can execute remote jobs on any host.

If LSF_ROOT_REX is set, root can also execute remote jobs with the same permission test as for normal users.

Default Undefined

See also "LSF_ROOT_REX" on page 567, "LSF_AUTH" on page 537 in "lsf.conf"

LSF_USER_DOMAIN

Syntax LSF_USER_DOMAIN=*domain_name* | **.**

Description Set during LSF installation or setup. If you modify this parameter in an existing cluster, you probably have to modify passwords and configuration files also.

Windows or mixed UNIX-Windows clusters only.

Enables default user mapping, and specifies the LSF user domain. The period (`.`) specifies local accounts, not domain accounts.

- A user name specified without a domain is interpreted (on a Windows host) as belonging to the LSF user domain
- A user name specified with the domain name of the LSF user domain is not valid
- In a mixed cluster, this parameter defines a 2-way, 1:1 user map between UNIX user accounts and Windows user accounts belonging to the specified domain, as long as the accounts have the same user name. This means jobs submitted by the Windows user account can run on a UNIX host, and jobs submitted by the UNIX account can run on any Windows host that is available to the Windows user account.

If this parameter is undefined, the default user mapping is not enabled. You can still configure user mapping at the user or system level. User account mapping is required to run cross-platform jobs in a UNIX-Windows mixed cluster.

Where Defined `lsf.conf`

Default

- If you upgrade from LSF 4.0.1 or earlier, the default is the existing LSF user domain.
- For a new, Windows-only cluster, this parameter is undefined (no LSF user domain, no default user mapping).
- For a new, mixed UNIX-Windows cluster, the default is the domain that the Windows installation account belongs to. This can be modified during LSF installation.

P A R T III

Configuration Files

- "bld.license.acct" on page 295
- "cshrc.lsf and profile.lsf" on page 297
- "hosts" on page 303
- "install.config" on page 307
- "lim.acct" on page 313
- "lsb.acct" on page 315
- "lsb.events" on page 323
- "lsb.hosts" on page 353
- "lsb.modules" on page 367
- "lsb.params" on page 373
- "lsb.queues" on page 399
- "lsb.resources" on page 433
- "lsb.serviceclasses" on page 457
- "lsb.users" on page 465
- "lsf.acct" on page 475
- "lsf.cluster" on page 479
- "lsf.cluster_name.license.acct" on page 499
- "lsf.conf" on page 503
- "lsf.licensescheduler" on page 577
- "lsf.shared" on page 599
- "lsf.sudoers" on page 607
- "lsf.task" on page 617
- "setup.config" on page 623
- "slave.config" on page 627
- "win_install.config" on page 633

bld.license.acct

The `bld.license.acct` file is the license and accounting file for LSF License Scheduler.

Contents

◆ "bld.license.acct Structure" on page 296

bld.license.acct Structure

The license accounting log file is an ASCII file with one record per line. The fields of a record are separated by blanks. LSF License Scheduler adds a new record to the file every hour.

File properties

Location The default location of this file is `LSF_SHAREDIR/db`. Use `LSF_LICENSE_ACCT_PATH` in `lsf.conf` to specify another location.

Owner The primary LSF License Scheduler admin is the owner of this file.

Permissions `-rw-r--r--`

Records and fields

The fields in order of occurence are as follows:

`timestamp (%d)`

Time stamp of the logged event (in seconds since the epoch).

`type (%s)`

The LSF product type. For LSF License Scheduler, this is `LICENSE_SCHEDULER`.

`version (%s)`

The version of the LSF License Scheduler product.

`value (%d)`

The total number of tokens that LSF License Scheduler is using.

`status (%s)`

The results of the license usage check. The valid values are as follows:

- `OK`

  Token usage is less than the currently licensed amount
- `OVERUSE`

  Token usage is more than the currently licensed amount

`hash (%s)`

Line encryption used to authenticate the record.

Example record Format

```
1107961731 LICENSE_SCHEDULER 6.10 0 OK 335a33c2bd9c9428140a61e57bd06da02b623a42
1107961792 LICENSE_SCHEDULER 6.10 2 OK 58e45b891f371811edfcceb6f5270059a74ee31a
1126639979 LICENSE_SCHEDULER 6.2 0 5 OK b3efd43ee28346f2d125b445fd16aa96875da35
1126640028 LICENSE_SCHEDULER 6.2 6 5 OVERUSE 2865775920372225fa7f8ed4b9a8eb2b15
```

SEE ALSO

- LSF_LOGDIR in `lsf.conf`
- LSF_LICENSE_ACCT_PATH in `lsf.conf`
- `lsf.cluster_name.license.acct`

cshrc.lsf and profile.lsf

Contents

- "About cshrc.lsf and profile.lsf" on page 298
- "LSF Environment Variables Set by cshrc.lsf and profile.lsf" on page 301

About cshrc.lsf and profile.lsf

The user environment shell files `cshrc.lsf` and `profile.lsf` set the LSF operating environment on a Platform LSF host. They define machine-dependent paths to LSF commands and libraries as environment variables:

- `cshrc.lsf` sets the C shell (`csh` or `tcsh`) user environment for LSF commands and libraries
- `profile.lsf` sets and exports the Bourne shell/Korn shell (`sh`, `ksh`, or `bash`) user environment for LSF commands and libraries

LSF Administrators should make sure that `cshrc.lsf` or `profile.lsf` are available for users to set the LSF environment variables correctly for the host type running LSF.

Location

`cshrc.lsf` and `profile.lsf` are created by `lsfinstall` during installaton. After installation, they are located in LSF_CONFDIR (`LSF_TOP/conf/`).

Format

`cshrc.lsf` and `profile.lsf` are conventional UNIX shell scripts:

- `cshrc.lsf` runs under `/bin/csh`
- `profile.lsf` runs under `/bin/sh`

What cshrc.lsf and profile.lsf do

`cshrc.lsf` and `profile.lsf` determine the binary type (BINARY_TYPE) of the host and set environment variables for the paths to the following machine-dependent LSF directories, according to the LSF version (LSF_VERSION) and the location of the top-level installation directory (LSF_TOP) defined at installation:

- LSF_BINDIR
- LSF_SERVERDIR
- LSF_LIBDIR
- XLSF_UIDDIR

`cshrc.lsf` and `profile.lsf` also set the following user environment variables:

- LSF_ENVDIR
- LD_LIBRARY_PATH
- PATH to include the paths to:
  - LSF_BINDIR
  - LSF_SERVERDIR
- MANPATH to include the path to the LSF man pages

Setting up the LSF environment with cshrc.lsf and profile.lsf

Before using LSF, you must set up the LSF execution environment.

After logging on to an LSF host, use one of the following shell environment files to set your LSF environment:

- For example, in `csh` or `tcsh`:

  `%` **`source /usr/share/lsf/lsf_62/conf/cshrc.lsf`**

- For example, in `sh`, `ksh`, or `bash`:

  `$` **`. /usr/share/lsf/lsf_62/conf/profile.lsf`**

Making your cluster available to users with cshrc.lsf and profile.lsf

To set up the LSF user environment, run one of the following two shell files:

- `LSF_CONFDIR/cshrc.lsf` (for `csh`, `tcsh`)
- `LSF_CONFDIR/profile.lsf` (for `sh`, `ksh`, or `bash`)

LSF administrators should make sure all LSF users include one of these files at the end of their own `.cshrc` or `.profile` file, or run one of these two files before using LSF.

For csh or tcsh

Add `cshrc.lsf` to the end of the `.cshrc` file for all users:

- Copy the `cshrc.lsf` file into `.cshrc`

OR

- Add a line similar to the following to the end of `.cshrc`:

  `source /usr/share/lsf/lsf_62/conf/cshrc.lsf`

After running `cshrc.lsf`, use `setenv` to see the environment variable settings. For example:

```
% setenv
PATH=/usr/share/lsf/lsf_62/6.2/sparc-sol7-
32/bin:/usr/share/lsf/lsf_62/6.2/sparc-sol7-
32/etc:/home/user1/bin:/local/private/user1/bin:/etc:/usr/etc:/usr/local/bin:/u
sr/local/sbin:/bin:/usr/bin:/usr/sbin:/opt/local/bin:/local/share/bin:/opt/gnu/
bin:/sbin:/usr/bin/X11:/usr/bsd:/usr/ucb:/local/bin/X11:/usr/hosts:/usr/openwin
/bin:/usr/ccs/bin:/usr/vue/bin:.
...
MANPATH=/usr/share/lsf/lsf_62/6.2/man:/home/user1/man:/opt/SUNWhpc/man:/usr/man
:/usr/local/man:/usr/softbench/man:/usr/openwin/man:/opt/SUNWmotif/man:/opt/ans
ic/share/man:/opt/hpnp/man:/usr/share/man:/usr/share/catman
...
LSF_BINDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/bin
LSF_SERVERDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/etc
LSF_LIBDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib
LD_LIBRARY_PATH=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib
XLSF_UIDDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib/uid
LSF_ENVDIR=/usr/share/lsf/lsf_62/conf
```

For sh, ksh, or bash

Add `profile.lsf` to the end of the `.profile` file for all users:

- Copy the `profile.lsf` file into `.profile`

OR

- Add a line similar to following to the end of `.profile`:

  `. /usr/share/lsf/lsf_62/conf/profile.lsf`

After running `profile.lsf`, use the `set` command to see the environment variable settings. For example:

```
$ set
...
LD_LIBRARY_PATH=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib
LSF_BINDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/bin
LSF_ENVDIR=/usr/share/lsf/lsf_62/conf
LSF_LIBDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib
LSF_SERVERDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/etc
MANPATH=/usr/share/lsf/lsf_62/6.2/man:/home/user1/man:/opt/SUNWhpc/man:/usr/man
:/usr/local/man:/usr/softbench/man:/usr/openwin/man:/opt/SUNWmotif/man:/opt/ans
ic/share/man:/opt/hpnp/man:/usr/share/man:/usr/share/catman
PATH=/usr/share/lsf/lsf_62/6.2/sparc-sol7-
32/bin:/usr/share/lsf/lsf_62/6.2/sparc-sol7-
32/etc:/home/user1/bin:/local/private/user1/bin:/etc:/usr/etc:/usr/local/bin:/u
sr/local/sbin:/bin:/usr/bin:/usr/sbin:/opt/local/bin:/local/share/bin:/opt/gnu/
bin:/sbin:/usr/bin/X11:/usr/bsd:/usr/ucb:/local/bin/X11:/usr/hosts:/usr/openwin
/bin:/usr/ccs/bin:/usr/vue/bin:.
...
XLSF_UIDDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib/uid
...
```

cshrc.lsf and profile.lsf on dynamically added LSF slave hosts

Dynamically added LSF hosts that will not be master candidates are *slave hosts*. Each dynamic slave host has its own LSF binaries and local `lsf.conf` and shell environment scripts (`cshrc.lsf` and `profile.lsf`).

LSF Environment Variables Set by cshrc.lsf and profile.lsf

LSF_BINDIR

Syntax **LSF_BINDIR=***dir*

Description Directory where LSF user commands are installed.

Examples

- Set in csh and tcsh by cshrc.lsf:

  ```
  setenv LSF_BINDIR /usr/share/lsf/lsf_62/6.2/sparc-sol7-
  32/bin
  ```

- Set and exported in sh, ksh, or bash by profile.lsf:

  ```
  LSF_BINDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/bin
  ```

Values

- In cshrc.lsf for csh and tcsh:

  ```
  setenv LSF_BINDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/bin
  ```

- Set and exported in profile.lsf for sh, ksh, or bash:

  ```
  LSF_BINDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/bin
  ```

LSF_ENVDIR

Syntax **LSF_ENVDIR=***dir*

Description Directory containing the lsf.conf file.

By default, lsf.conf is installed by creating a shared copy in LSF_CONFDIR and adding a symbolic link from /etc/lsf.conf to the shared copy. If LSF_ENVDIR is set, the symbolic link is installed in LSF_ENVDIR/lsf.conf.

The lsf.conf file is a global environment configuration file for all LSF services and applications. The LSF default installation places the file in LSF_CONFDIR.

Examples

- Set in csh and tcsh by cshrc.lsf:

  ```
  setenv LSF_ENVDIR /usr/share/lsf/lsf_62/conf
  ```

- Set and exported in sh, ksh, or bash by profile.lsf:

  ```
  LSF_ENVDIR=/usr/share/lsf/lsf_62/conf
  ```

Values

- In cshrc.lsf for csh and tcsh:

  ```
  setenv LSF_ENVDIR $LSF_TOP/conf
  ```

- Set and exported in profile.lsf for sh, ksh, or bash:

  ```
  LSF_DIR=$LSF_TOP/conf
  ```

LSF_LIBDIR

Syntax **LSF_LIBDIR=***dir*

Description Directory where LSF libraries are installed. Library files are shared by all hosts of the same type.

Examples

- Set in csh and tcsh by cshrc.lsf:

  ```
  setenv LSF_LIBDIR /usr/share/lsf/lsf_62/6.2/sparc-sol7-
  32/lib
  ```

- Set and exported in `sh`, `ksh`, or `bash` by `profile.lsf`:

  `LSF_LIBDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib`

Values

- In `cshrc.lsf` for `csh` and `tcsh`:

  `setenv LSF_LIBDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib`

- Set and exported in `profile.lsf` for `sh`, `ksh`, or `bash`:

  `LSF_LIBDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib`

LSF_SERVERDIR

Syntax **`LSF_SERVERDIR=`***dir*

Description Directory where LSF server binaries and shell scripts are installed.

These include `lim`, `res`, `nios`, `sbatchd`, `mbatchd`, and `mbschd`. If you use `elim`, `eauth`, `eexec`, `esub`, etc, they are also installed in this directory.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:

  `setenv LSF_SERVERDIR /usr/share/lsf/lsf_62/6.2/sparc-sol7-32/etc`

- Set and exported in `sh`, `ksh`, or `bash` by `profile.lsf`:

  `LSF_SERVERDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/etc`

Values

- In `cshrc.lsf` for `csh` and `tcsh`:

  `setenv LSF_SERVERDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/etc`

- Set and exported in `profile.lsf` for `sh`, `ksh`, or `bash`:

  `LSF_SERVERDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/etc`

XLSF_UIDDIR

Syntax **`XLSF_UIDDIR=`***dir*

Description (UNIX only) Directory where Motif User Interface Definition files are stored.

These files are platform-specific.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:

  `setenv XLSF_UIDDIR /usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib/uid`

- Set and exported in `sh`, `ksh`, or `bash` by `profile.lsf`:

  `XLSF_UIDDIR=/usr/share/lsf/lsf_62/6.2/sparc-sol7-32/lib/uid`

Values

- In `cshrc.lsf` for `csh` and `tcsh`:

  `setenv XLSF_UIDDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib/uid`

- Set and exported in `profile.lsf` for `sh`, `ksh`, or `bash`:

  `XLSF_UIDDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib/uid`

SEE ALSO

`lsfinstall`(8), `install.config`(5), `lsf.cluster`(5), `lsf.conf`(5), `lsf.sudoers`(5), `slave.config`(5)

hosts

For hosts with multiple IP addresses and different official host names configured at the system level, this file associates the host names and IP addresses in LSF.

By default, LSF assumes each host in the cluster:

- Has a unique "official" host name
- Can resolve its IP address from its name
- Can resolve its official name from its IP address

Hosts with only one IP address, or hosts with multiple IP addresses that already resolve to a unique official host name should not be configured in this file: they are resolved using the default method for your system (for example, local configuration files like `/etc/hosts` or through DNS.)

The LSF `hosts` file is used in environments where:

- Machines in cluster have multiple network interfaces and cannot be set up in the system with a unique official host name
- DNS is slow or not configured properly
- Machines have special topology requirements; for example, in HPC systems where it is desirable to map multiple actual hosts to a single "head end" host

The LSF `hosts` file is not installed by default. It is usually located in the directory specified by LSF_CONFDIR. The format of `LSF_CONFDIR/hosts` is similar to the format of the `/etc/hosts` file on all UNIX machines.

hosts File Structure

One line for each IP address, consisting of the IP address, followed by the official host name, optionally followed by host aliases, all separated by spaces or tabs. Each line has the form:

`ip_address official_name [alias [alias ...]]`

Use consecutive lines for IP addresses belonging to the same host. You can assign different aliases to different addresses.

Use a pound sign (#) to indicate a comment (the rest of the line is not read by LSF). Do not use `#if` as this is reserved syntax for time-based configuration.

A call to `gethostbyname(3N)` returns a `hostent` structure containing the union of all addresses and aliases from each line containing a matching official host name or alias.

IP Address

Written using the conventional dotted decimal notation (`nnn.nnn.nnn.nnn`) and interpreted using the `inet_addr` routine from the Internet address manipulation library, `inet(3N)`.

Official Host Name

The official host name. Single character names are not allowed.

Specify `-GATEWAY` or `-GW` as part of the host name if the host serves as a GATEWAY.

Specify `-TAC` as the last part of the host name if the host is a TAC and is a DoD host.

Specify the host name in the format defined in Internet RFC 952, which states:

> A "name" (Net, Host, Gateway, or Domain name) is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). Periods are only allowed when they serve to delimit components of "domain style names". (See RFC 921, "Domain Name System Implementation Schedule", for background). No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be an alpha character. The last character must not be a minus sign or a period.

RFC 952 has been modified by RFC 1123 to relax the restriction on the first character being a digit.

For maximum interoperability with the Internet, you should use host names no longer than 24 characters for the host portion (exclusive of the domain component).

Aliases

Optional. Aliases to the host name.

The default host file syntax

ip_address official_name [*alias* [*alias* ...]]

is powerful and flexible, but it is difficult to configure in systems where a single host name has many aliases, and in multihomed host environments.

In these cases, the `hosts` file can become very large and unmanageable, and configuration is prone to error.

The syntax of the LSF `hosts` file supports host name ranges as aliases for an IP address. This simplifies the host name alias specification.

To use host name ranges as aliases, the host names must consist of a fixed node group name prefix and node indices, specified in a form like:

host_name[*index_x*-*index_y*, *index_m*, *index_a*-*index_b*]

For example:

```
atlasD0[0-3,4,5-6, ...]
```

is equivalent to:

```
atlasD0[0-6, ...]
```

The node list does not need to be a continuous range (some nodes can be configured out). Node indices can be numbers or letters (both upper case and lower case).

For example, some systems map internal compute nodes to single LSF host names. A host file might contains 64 lines, each specifying an LSF host name and 32 node names that correspond to each LSF host:

```
...
177.16.1.1 atlasD0 atlas0 atlas1 atlas2 atlas3 atlas4 ... atlas31
177.16.1.2 atlasD1 atlas32 atlas33 atlas34 atlas35 atlas36 ... atlas63
...
```

In the new format, you still map the nodes to the LSF hosts, so the number of lines remains the same, but the format is simplified because you only have to specify ranges for the nodes, not each node individually as an alias:

```
...
177.16.1.1 atlasD0 atlas[0-31]
177.16.1.2 atlasD1 atlas[32-63]
...
```

Example hosts file

```
192.168.1.1 hostA hostB
192.168.2.2 hostA hostC host-C
```

In this example, `hostA` has 2 IP addresses and 3 aliases. The alias `hostB` specifies the first address, and the aliases `hostC` and `host-C` specify the second address. LSF uses the official host name, `hostA`, to identify that both IP addresses belong to the same host.

install.config

Contents

- “About install.config” on page 308
- “Parameters” on page 309

About install.config

The install.config file contains options for Platform LSF installation and configuration. Use lsfinstall -f install.config to install LSF using the options specified in install.config.

Template location

A template install.config is included in the installation script tar file lsf6.2_lsfinstall.tar.Z and is located in the lsf6.2_lsfinstall directory created when you uncompress and extract installation script tar file. Edit the file and uncomment the options you want in the template file. Replace the example values with your own settings to specify the options for your new installation.

Important The sample values in the install.config template file are examples only. They are not default installation values.

After installation, the install.config containing the options you specified is located in LSF_TOP/6.2/install/.

Format

Each entry in install.config has the form:

```
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each NAME even if no value follows and there should be no spaces around the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Blank lines and lines starting with a pound sign (#) are ignored.

Parameters

- "LSF_ADD_SERVERS"
- "LSF_ADD_CLIENTS"
- "LSF_ADMINS"
- "LSF_CLUSTER_NAME"
- "LSF_DYNAMIC_HOST_WAIT_TIME"
- "LSF_LICENSE"
- "LSF_MASTER_LIST"
- "LSF_QUIET_INST"
- "LSF_TARDIR"
- "LSF_TOP"
- "ENABLE_HPC_INST"

LSF_ADD_SERVERS

Syntax **LSF_ADD_SERVERS="***host_name* [*host_name*...]**"**

Description Lists the hosts in the cluster to be set up as server hosts. The first host in the list becomes the master host in `lsf.cluster.`*cluster_name*.

Valid Values Any valid host name

Example `LSF_ADD_SERVERS="hosta hostb hostc hostd"`

`hosta` is the master host.

Default The local host where `lsfinstall` is running

See Also LSF_ADD_CLIENTS

LSF_ADD_CLIENTS

Syntax **LSF_ADD_CLIENTS="***host_name* [*host_name*...]**"**

Description Lists the hosts in the cluster to be set up as client-only hosts.

After installation, you must manually edit `lsf.cluster.`*cluster_name* to include the correct host model and type of each static client listed in LSF_ADD_CLIENTS. This will enable automatic host type and model detection when the client host LIM starts.

Valid Values Any valid host name

Example `LSF_ADD_CLIENTS="hoste hostf"`

Default None

See Also LSF_ADD_SERVERS

LSF_ADMINS

Syntax **LSF_ADMINS="***user_name* [*user_name* ...]**"**

Description Lists the LSF administrators. The first user account name in the list is the primary LSF administrator in `lsf.cluster.`*cluster_name*.

The LSF administrator accounts must exist on all hosts in the cluster before installing LSF

The primary LSF administrator account is typically named `lsfadmin`. It owns the LSF configuration files and log files for job events. It also has permission to reconfigure LSF and to control batch jobs submitted by other users. It typically does not have authority to start LSF daemons. Unless an `lsf.sudoers` file exists to grant LSF administrators permission, only root has permission to start LSF daemons.

CAUTION **You should *not* configure the root account as the primary LSF administrator.**

Valid Values User accounts for LSF administrators must exist on all hosts in the cluster before running `lsfinstall`.

Example `LSF_ADMINS="lsfadmin user1 user2"`

Default None—required variable

LSF_CLUSTER_NAME

Syntax **LSF_CLUSTER_NAME="***cluster_name***"**

Description Defines the name of the cluster.

Valid Values Any alphanumeric string containing no more than 39 characters. The name cannot contain white spaces.

Do not use the name of any host, user, or user group as the name of your cluster.

Example `LSF_CLUSTER_NAME="cluster1"`

Default None—required variable

LSF_DYNAMIC_HOST_WAIT_TIME

Syntax **LSF_DYNAMIC_HOST_WAIT_TIME=***time_seconds*

Description Defines the period of time from startup for dynamic slave LIMs (hosts) to wait for an acknowledgement from the master LIM. This signals to the dynamic host that it is already in the cluster and therefore does not need to be added. If it does not receive this acknowledgement, the dynamic host sends a request to the master LIM to add it to the cluster.

To enable dynamically added hosts, you must define both LSF_DYNAMIC_HOST_WAIT_TIME in lsf.conf, and LSF_HOST_ADDR_RANGE in lsf.cluster.*cluster_name*.

Recommended value Up to 60 seconds for every 1000 hosts in the cluster, for a maximum of 15 minutes. Selecting a smaller value will result in a quicker response time for new hosts at the expense of an increased load on the master LIM.

Example `LSF_DYNAMIC_HOST_WAIT_TIME=60`

Hosts will wait 60 seconds from startup to receive an acknowledgement from the master LIM. If it does not receive the acknowledgement within the 60 seconds, it will send a request for the master LIM to add it to the cluster.

Default INFINIT_INT (the host will never send a request to the master LIM)

LSF_LICENSE

Syntax **LSF_LICENSE="/***path***/***license_file***"**

Description Full path to the name of the LSF license file. You must have a valid license file to install LSF.

If you do not specify LSF_LICENSE, or `lsfinstall` cannot find a valid license file in the default location, `lsfinstall` exits.

Recommended Value /*path*/`license.dat`

Example `LSF_LICENSE="/usr/share/lsf_distrib/liscense.dat"`

Default /*current_directory*/`license.dat`

LSF_MASTER_LIST

Syntax **LSF_MASTER_LIST="***host_name* [*host_name* ...]**"**

Description Optional. Defines a list of hosts that are candidates to become the master host for the cluster. Listed hosts must be defined as servers in LSF_ADD_SERVERS.

Required for dynamic host configuration. To dynamically add or remove hosts, you must specify a list of candidate master hosts. If you do not need to add or remove hosts dynamically, you can leave this parameter undefined during new installation or upgrade.

Specify a list of host names two ways:

- Host names separated by spaces
- Name of a file containing a list of host names, one host per line.

Master candidate hosts should share LSF configuration and binaries.

Valid Values Any valid host name

Examples

- List of host names:

  `LSF_MASTER_LIST="hosta hostb hostc hostd"`

- Host list file:

  `LSF_MASTER_LIST=:lsf_master_list`

  The file `lsf_master_list` contains a list of hosts:

```
hosta
hostb
hostc
hostd
```

Default None—optional variable

LSF_QUIET_INST

Syntax **LSF_QUIET_INST="y | n"**

Description Do not display `lsfinstall` messages.

Example `LSF_QUIET_INST="y"`

Default Display all messages. (LSF_QUIET_INST="n")

LSF_TARDIR

Syntax **LSF_TARDIR="/*path*"**

Description Full path to the directory containing the LSF distribution tar files.

Example `LSF_TARDIR="/usr/share/lsf_distrib"`

Default The parent directory of the current working directory where `lsfinstall` is running (`../current_directory`)

LSF_TOP

Syntax **LSF_TOP="/*path*"**

Description Top-level LSF installation directory.

Valid Values Must be an absolute path to a shared directory that is accessible to all LSF hosts. Cannot be the root directory (`/`).

Recommended Value The file system containing LSF_TOP must have enough disk space for all host types (approximately 300 MB per host type).

Example `LSF_TOP="/usr/share/lsf"`

Default None—required variable

ENABLE_HPC_INST

Syntax **ENABLE_HPC_INST=Y | N**

Description Optional. Enables Platform LSF HPC installation, and adds the `Platform_HPC` license keyword to the PRODUCTS line of `lsf.cluster.cluster_name` if it is not already there.

Default N (Platform LSF HPC is not configured.)

SEE ALSO

`lsfinstall`(8), `lsf.cluster`(5), `lsf.sudoers`(5), `slave.config`(5)

lim.acct

The `lim.acct` file is the log file for Load Information Manager (LIM). Produced by `lsmon`, `lim.acct` contains host load information collected and distributed by LIM.

Contents

◆ "lim.acct Structure" on page 314

lim.acct Structure

The first line of `lim.acct` contains a list of load index names separated by spaces. This list of load index names can be specified in the `lsmon` command line. The default list is "`r15s r1m r15m ut pg ls it swp mem tmp`". Subsequent lines in the file contain the host's load information at the time the information was recorded.

Fields

Fields are ordered in the following sequence:

`time (%ld)`

The time when the load information is written to the log file

`host name (%s)`

The name of the host.

`status of host (%d)`

An array of integers. The first integer marks the operation status of the host. Additional integers are used as a bit map to indicate load status of the host. An integer can be used for 32 load indices. If the number of user defined load indices is not more than 21, only one integer is used for both built-in load indices and external load indices. See the `hostload` structure in `ls_load(3)` for the description of these fields.

`indexvalue (%f)`

A sequence of load index values. Each value corresponds to the index name in the first line of `lim.acct`. The order in which the index values are listed is the same as the order of the index names.

SEE ALSO

Related Topics `lsmon`(1), `lsload`(1)

Files None

lsb.acct

The lsb.acct file is the batch job log file of LSF. The master batch daemon (see mbatchd(8)) generates a record for each job completion or failure. The record is appended to the job log file lsb.acct.

The file is located in LSB_SHAREDIR/*cluster_name*/logdir, where LSB_SHAREDIR must be defined in lsf.conf(5) and *cluster_name* is the name of the LSF cluster, as returned by lsid(1). See mbatchd(8) for the description of LSB_SHAREDIR.

The bacct command uses the current lsb.acct file for its output.

Contents

- "lsb.acct Structure" on page 316

lsb.acct Structure

The job log file is an ASCII file with one record per line. The fields of a record are separated by blanks. If the value of some field is unavailable, a pair of double quotation marks ("") is logged for character string, 0 for time and number, and -1 for resource usage.

Configuring automatic archiving

The following parameters in `lsb.params` affect how records are logged to `lsb.acct`:

- ACCT_ARCHIVE_AGE=*days*

  Enables automatic archiving of LSF accounting log files, and specifies the archive interval. LSF archives the current log file if the length of time from its creation date exceeds the specified number of days.

  By default there is no limit to the age of lsb.acct.
- ACCT_ARCHIVE_SIZE=*kilobytes*

  Enables automatic archiving of LSF accounting log files, and specifies the archive threshold. LSF archives the current log file if its size exceeds the specified number of kilobytes.

  By default, there is no limit to the size of lsb.acct.
- ACCT_ARCHIVE_TIME=*hh*:*mm*

  Enables automatic archiving of LSF accounting log file lsb.acct, and specifies the time of day to archive the current log file.

  By default, no time is set for archiving lsb.acct.
- MAX_ACCT_ARCHIVE_FILE=*integer*

  Enables automatic deletion of archived LSF accounting log files and specifies the archive limit.

  By default, `lsb.acct.`*n* files are not automatically deleted.

Records and fields

The fields of a record are separated by blanks. The first string of an event record indicates its type. The following types of events are recorded:

- JOB_FINISH
- EVENT_ADRSV_FINISH

JOB_FINISH

A job has finished.

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, older daemons and commands (pre-LSF Version 6.0) cannot recognize the `lsb.acct` file format.

The fields in order of occurrence are:

`Event type (%s)`

Which is “JOB_FINISH"

`Version Number (%s)`

Version number of the log file format

`Event Time (%d)`

Time the event was logged (in seconds since the epoch)

`jobId (%d)`

ID for the job

`userId (%d)`

UNIX user ID of the submitter

`options (%d)`

Bit flags for job processing

`numProcessors (%d)`

Number of processors initially requested for execution

`submitTime (%d)`

Job submission time

`beginTime (%d)`

Job start time – the job should be started at or after this time

`termTime (%d)`

Job termination deadline – the job should be terminated by this time

`startTime (%d)`

Job dispatch time – time job was dispatched for execution

`userName (%s)`

User name of the submitter

`queue (%s)`

Name of the job queue to which the job was submitted

`resReq (%s)`

Resource requirement specified by the user

`dependCond (%s)`

Job dependency condition specified by the user

`preExecCmd (%s)`

Pre-execution command specified by the user

`fromHost (%s)`

Submission host name

`cwd (%s)`

Current working directory

`inFile (%s)`

Input file name (%s)

`outFile (%s)`

output file name

`errFile (%s)`

Error output file name

`jobFile (%s)`

Job script file name

`numAskedHosts (%d)`

Number of host names to which job dispatching will be limited

`askedHosts (%s)`

List of host names to which job dispatching will be limited (%s for each); nothing is logged to the record for this value if the last field value is 0. If there is more than one host name, then each additional host name will be returned in its own field

`numExHosts (%d)`

Number of processors used for execution

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of .hosts listed in the `execHosts` field. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`execHosts (%s)`

List of execution host names (%s for each); nothing is logged to the record for this value if the last field value is 0

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`jStatus (%d)`

Job status. The number 32 represents EXIT, 64 represents DONE

`hostFactor (%f)`

CPU factor of the first execution host

`jobName (%s)`

Job name

`command (%s)`

Complete batch job command specified by the user

`lsfRusage (%f)`

The following fields contain resource usage information for the job (see `getrusage`(2)). If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

`ru_utime (%f)`

User time used

`ru_stime (%f)`

System time used

`ru_maxrss (%f)`

Maximum shared text size

`ru_ixrss (%f)`

Integral of the shared text size over time (in KB seconds)

`ru_ismrss (%f)`

Integral of the shared memory size over time (valid only on Ultrix)

`ru_idrss (%f)`

Integral of the unshared data size over time

`ru_isrss (%f)`

Integral of the unshared stack size over time

`ru_minflt (%f)`

Number of page reclaims

`ru_majflt (%f)`

Number of page faults

`ru_nswap (%f)`

Number of times the process was swapped out

`ru_inblock (%f)`

Number of block input operations

`ru_oublock (%f)`

Number of block output operations

`ru_ioch (%f)`

Number of characters read and written (valid only on HP-UX)

`ru_msgsnd (%f)`

Number of System V IPC messages sent

`ru_msgrcv (%f)`

Number of messages received

`ru_nsignals (%f)`

Number of signals received

`ru_nvcsw (%f)`

Number of voluntary context switches

`ru_nivcsw (%f)`

Number of involuntary context switches

`ru_exutime (%f)`

Exact user time used (valid only on ConvexOS) `mailUser (%s)`

Name of the user to whom job related mail was sent

`projectName (%s)`

LSF project name

`exitStatus (%d)`

UNIX exit status of the job

`maxNumProcessors (%d)`

Maximum number of processors specified for the job

`loginShell (%s)`

Login shell used for the job

`timeEvent (%s)`

Time event string for the job - JobScheduler only

`idx (%d)`

Job array index

`maxRMem (%d)`

Maximum resident memory usage in KBytes of all processes in the job

`maxRSwap (%d)`

Maximum virtual memory usage in KBytes of all processes in the job

`inFileSpool (%s)`

Spool input file

`commandSpool (%s)`

Spool command file

`rsvId %s`

Advance reservation ID; for example, "`user2#0`"

`sla (%s)`

SLA service class name under which the job runs

`exceptMask (%d)`

Job exception handling

Values:

- `J_EXCEPT_OVERRUN 0x02`
- `J_EXCEPT_UNDERUN 0x04`
- `J_EXCEPT_IDLE 0x80`

`additionalInfo (%s)`

Placement information of HPC jobs

`exitInfo (%d)`

Job termination reason, see `<lsbatch/lsbatch.h>`

`warningAction (%s)`

Job warning action

`warningTimePeriod (%d)`

Job warning time period in seconds

`chargedSAAP (%s)`

SAAP charged to a job

`licenseProject (%s)`

LSF License Scheduler project name

EVENT_ADRSV_FINISH

An advance reservation has expired. The fields in order of occurrence are:

`Event type (%s)`

Which is "EVENT_ADRSV_FINISH"

`Version Number (%s)`

Version number of the log file format

`Event Logging Time (%d)`

Time the event was logged (in seconds since the epoch); for example, "`1038942015`"

`Reservation Creation Time (%d)`

Time the advance reservation was created (in seconds since the epoch); for example, "`1038938898`"

`Reservation Type (%d)`

Type of advance reservation request:

- User reservation (RSV_OPTION_USER, defined as 0x001)
- User group reservation (RSV_OPTION_GROUP, defined as 0x002)
- System reservation (RSV_OPTION_SYSTEM, defined as 0x004)
- Recurring reservation (RSV_OPTION_RECUR, defined as 0x008)

For example, "`9`" is a recurring reservation created for a user.

`Creator ID (%d)`

UNIX user ID of the reservation creator; for example, "`30408`"

`Reservation ID (rsvId %s)`

For example, "`user2#0`"

`User Name (%s)`

User name of the reservation user; for example, "`user2`"

`Time Window (%s)`

Time window of the reservation:

- One-time reservation in seconds since the epoch; for example, "`1033761000-1033761600`"
- Recurring reservation; for example, "`17:50-18:00`"

`Creator Name (%s)`

User name of the reservation creator; for example, "`user1`"

`Duration (%d)`

Duration of the reservation, in hours, minutes, seconds; for example, "`600`" is 6 hours, 0 minutes, 0 seconds

`Number of Resources (%d)`

Number of reserved resource pairs in the resource list; for example "`2`" indicates 2 resource pairs ("`hostA 1 hostB 1`")

`Host Name (%s)`

Reservation host name; for example, "`hostA`"

`Number of CPUs (%d)`

Number of reserved CPUs; for example "`1`"

SEE ALSO

Related topics `lsb.events`(5), `lsb.params`(5), `lsf.conf`(5), `mbatchd`(8), `bacct`(1), `brsvadd`(8), `brsvs`(1), `bsub`(1), `lsid`(1)

Files `$LSB_SHAREDIR/`*`cluster_name`*`/logdir/lsb.acct`

lsb.events

The LSF batch event log file `lsb.events` is used to display LSF batch event history and for `mbatchd` failure recovery.

Whenever a host, job, or queue changes status, a record is appended to the event log file. The file is located in `LSB_SHAREDIR/`*`cluster_name`*`/logdir`, where LSB_SHAREDIR must be defined in `lsf.conf(5)` and *`cluster_name`* is the name of the LSF cluster, as returned by `lsid`(1). See `mbatchd`(8) for the description of LSB_SHAREDIR.

The `bhist` command searches the most current `lsb.events` file for its output.

Contents

- "lsb.events Structure" on page 324

lsb.events Structure

The event log file is an ASCII file with one record per line. For the `lsb.events` file, the first line has the format "# <history seek position>", which indicates the file position of the first history event after log switch. For the `lsb.events.#` file, the first line has the format "# <timestamp of most recent event>", which gives the timestamp of the recent event in the file.

Limiting the size of lsb.events

Use MAX_JOB_NUM in lsb.params to set the maximum number of finished jobs whose events are to be stored in the `lsb.events` log file.

Once the limit is reached, `mbatchd` starts a new event log file. The old event log file is saved as `lsb.events.`*n*, with subsequent sequence number suffixes incremented by 1 each time a new log file is started. Event logging continues in the new `lsb.events` file.

Records and fields

The fields of a record are separated by blanks. The first string of an event record indicates its type. The following types of events are recorded:

- JOB_NEW
- JOB_FORWARD
- JOB_ACCEPT
- JOB_START
- JOB_START_ACCEPT
- JOB_STATUS
- JOB_SWITCH
- JOB_MOVE
- QUEUE_CTRL
- HOST_CTRL
- MBD_START
- MBD_DIE
- UNFULFILL
- LOAD_INDEX
- JOB_SIGACT
- MIG
- JOB_MODIFY2
- JOB_SIGNAL
- JOB_EXECUTE
- JOB_REQUEUE
- JOB_CLEAN
- JOB_EXCEPTION
- JOB_EXT_MSG
- JOB_ATTA_DATA
- JOB_CHUNK
- SBD_UNREPORTED_STATUS

- PRE_EXEC_START
- JOB_FORCE

JOB_NEW

A new job has been submitted. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`userId (%d)`

UNIX user ID of the submitter

`options (%d)`

Bit flags for job processing

`numProcessors (%d)`

Number of processors requested for execution

`submitTime (%d)`

Job submission time

`beginTime (%d)`

Start time – the job should be started on or after this time

`termTime (%d)`

Termination deadline – the job should be terminated by this time (%d)

`sigValue (%d)`

Signal value

`chkpntPeriod (%d)`

Checkpointing period

`restartPid (%d)`

Restart process ID

`userName (%s)`

User name

`rLimits`

Soft CPU time limit (%d), see `getrlimit(2)`

`rLimits`

Soft file size limit (%d), see `getrlimit(2)`

`rLimits`

Soft data segment size limit (%d), see `getrlimit(2)` `rLimits`

Soft stack segment size limit (%d), see `getrlimit(2)`

`rLimits`

Soft core file size limit (%d), see `getrlimit(2)`

`rLimits`

Soft memory size limit (%d), see `getrlimit(2)`

`rLimits`

Reserved (%d)

`rLimits`

Reserved (%d)

`rLimits`

Reserved (%d)

`rLimits`

Soft run time limit (%d), see `getrlimit`(2)

`rLimits`

Reserved (%d)

`hostSpec (%s)`

Model or host name for normalizing CPU time and run time

`hostFactor (%f)`

CPU factor of the above host

`umask (%d)`

File creation mask for this job

`queue (%s)`

Name of job queue to which the job was submitted

`resReq (%s)`

Resource requirements

`fromHost (%s)`

Submission host name

`cwd (%s)`

Current working directory

`chkpntDir (%s)`

Checkpoint directory

`inFile (%s)`

Input file name

`outFile (%s)`

Output file name

`errFile (%s)`

Error output file name

`subHomeDir (%s)`

Submitter's home directory

`jobFile (%s)`

Job file name

`numAskedHosts (%d)`

Number of candidate host names

`askedHosts (%s)`

List of names of candidate hosts for job dispatching

`dependCond (%s)`

Job dependency condition

`preExecCmd (%s)`

Job pre-execution command

`timeEvent (%d)`

Time Event, for job dependency condition; specifies when time event ended

`jobName (%s)`

Job name

`command (%s)`

Job command

`nxf (%d)`

Number of files to transfer (%d)

`xf (%s)`

List of file transfer specifications

`mailUser (%s)`

Mail user name

`projectName (%s)`

Project name

`niosPort (%d)`

Callback port if batch interactive job

`maxNumProcessors (%d)`

Maximum number of processors

`schedHostType (%s)`

Execution host type

`loginShell (%s)`

Login shell

`userGroup (%s)`

User group

`exceptList (%s)`

Exception handlers for the job

`options2 (%d)`

Bit flags for job processing

`idx (%d)`

Job array index

`inFileSpool (%s)`

Spool input file

`commandSpool (%s)`

Spool command file

`jobSpoolDir (%s)`

Job spool directory

`userPriority (%d)`

User priority

`rsvId %s`

Advance reservation ID; for example, "`user2#0`"

`jobGroup (%s)`

The job group under which the job runs

`extsched (%s)`

External scheduling options

`warningAction (%s)`

Job warning action

`warningTimePeriod (%d)`

Job warning time period in seconds

`sla (%s)`

SLA service class name under which the job runs

`SLArunLimit (%d)`

Absolute run time limit of the job for SLA service classes

`licenseProject (%s)`

LSF License Scheduler project name

JOB_FORWARD

A job has been forwarded to a remote cluster (Platform MultiCluster only).

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, older daemons and commands (pre-LSF Version 6.0) cannot recognize the `lsb.events` file format.

The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`numReserHosts (%d)`

Number of reserved hosts in the remote cluster

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of .hosts listed in the `reserHosts` field. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`cluster (%s)`

Remote cluster name

`reserHosts (%s)`

List of names of the reserved hosts in the remote cluster

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`idx (%d)`

Job array index

JOB_ACCEPT

A job from a remote cluster has been accepted by this cluster. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID at the accepting cluster

`remoteJid (%d)`

Job ID at the submission cluster

`cluster (%s)`

Job submission cluster name

`idx (%d)`

Job array index

JOB_START

A job has been dispatched.

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, older daemons and commands (pre-LSF Version 6.0) cannot recognize the `lsb.events` file format.

The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`jStatus (%d)`

Job status, (4, indicating the RUN status of the job)

`jobPid (%d)`

Job process ID

`jobPGid (%d)`

Job process group ID

`hostFactor (%f)`

CPU factor of the first execution host

`numExHosts (%d)`

Number of processors used for execution

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of .hosts listed in the `execHosts` field. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`execHosts (%s)`

List of execution host names

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`queuePreCmd (%s)`

Pre-execution command

`queuePostCmd (%s)`

Post-execution command

`jFlags (%d)`

Job processing flags

`userGroup (%s)`

User group name

`idx (%d)`

Job array index

`additionalInfo (%s)`

Placement information of HPC jobs

JOB_START_ACCEPT

A job has started on the execution host(s). The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`jobPid (%d)`

Job process ID

`jobPGid (%d)`

Job process group ID

`idx (%d)`

Job array index

JOB_STATUS

The status of a job changed after dispatch. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`jStatus (%d)`

New status, see `<lsbatch/lsbatch.h>`

`reason (%d)`

Pending or suspended reason code, see `<lsbatch/lsbatch.h>`

`subreasons (%d)`

Pending or suspended subreason code, see `<lsbatch/lsbatch.h>`

`cpuTime (%f)`

CPU time consumed so far

`endTime (%d)`

Job completion time

`ru (%d)`

Resource usage flag

`lsfRusage (%s)`

Resource usage statistics, see `<lsf/lsf.h>`

`exitStatus (%d)`

Exit status of the job, see `<lsbatch/lsbatch.h>`

`idx (%d)`

Job array index

`exitInfo (%d)`

Job termination reason, see `<lsbatch/lsbatch.h>`

JOB_SWITCH

A job switched from one queue to another (`bswitch`). The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`userId (%d)`

UNIX user ID of the user invoking the command

`jobId (%d)`

Job ID

`queue (%s)`

Target queue name

`idx (%d)`

Job array index

`userName (%s)`

Name of the job submitter

JOB_MOVE

A job moved toward the top or bottom of its queue (`bbot` or `btop`). The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`userId (%d)`

UNIX user ID of the user invoking the command

`jobId (%d)`

Job ID

`position (%d)`

Position number

`base (%d)`

Operation code, (TO_TOP or TO_BOTTOM), see `<lsbatch/lsbatch.h>`

`idx (%d)`

Job array index

`userName (%s)`

Name of the job submitter

QUEUE_CTRL

A job queue has been altered. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`opCode (%d)`

Operation code), see `<lsbatch/lsbatch.h>`

`queue (%s)`

Queue name

`userId (%d)`

UNIX user ID of the user invoking the command

`userName (%s)`

Name of the user

`ctrlComments (%s)`

Administrator comment text from the `-C` option of `badmin` queue control commands `qclose`, `qopen`, `qact`, and `qinact`

HOST_CTRL

A batch server host changed status. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`opCode (%d)`

Operation code, see `<lsbatch/lsbatch.h>`

`host (%s)`

Host name

`userId (%d)`

UNIX user ID of the user invoking the command

`userName (%s)`

Name of the user

`ctrlComments (%s)`

Administrator comment text from the `-C` option of `badmin` host control commands `hclose` and `hopen`

MBD_START

The `mbatchd` has started. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`master (%s)`

Master host name

`cluster (%s)`

cluster name

`numHosts (%d)`

Number of hosts in the cluster

`numQueues (%d)`

Number of queues in the cluster

MBD_DIE

The `mbatchd` died. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`master (%s)`

Master host name

`numRemoveJobs (%d)`

Number of finished jobs that have been removed from the system and logged in the current event file

`exitCode (%d)`

Exit code from `mbatchd`

`ctrlComments (%s)`

Administrator comment text from the `-C` option of `badmin mbdrestart`

UNFULFILL

Actions that were not taken because the `mbatchd` was unable to contact the `sbatchd` on the job execution host. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`notSwitched (%d)`

Not switched: the `mbatchd` has switched the job to a new queue, but the `sbatchd` has not been informed of the switch

`sig (%d)`

Signal: this signal has not been sent to the job

`sig1 (%d)`

Checkpoint signal: the job has not been sent this signal to checkpoint itself

`sig1Flags (%d)`

Checkpoint flags, see `<lsbatch/lsbatch.h>`

`chkPeriod (%d)`

New checkpoint period for job

`notModified (%s)`

If set to true, then parameters for the job cannot be modified.

`idx (%d)`

Job array index

LOAD_INDEX

`mbatchd` restarted with these load index names (see `lsf.cluster`(5)). The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`nIdx (%d)`

Number of index names

`name (%s)`

List of index names

JOB_SIGACT

An action on a job has been taken. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`period (%d)`

Action period

`pid (%d)`

Process ID of the child `sbatchd` that initiated the action

`jstatus (%d)`

Job status

`reasons (%d)`

Job pending reasons

`flags (%d)`

Action flags, see `<lsbatch/lsbatch.h>`

`actStatus (%d)`

Action status:

1: Action started

2: One action preempted other actions

3: Action succeeded

4: Action Failed

`signalSymbol (%s)`

Action name, accompanied by actFlags

`idx (%d)`

Job array index

MIG

A job has been migrated (`bmig`). The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`numAskedHosts (%d)`

Number of candidate hosts for migration

`askedHosts (%s)`

List of names of candidate hosts

`userId (%d)`

UNIX user ID of the user invoking the command

`idx (%d)`

Job array index

`userName (%s)`

Name of the job submitter

JOB_MODIFY2

This is created when the `mbatchd` modifies a previously submitted job with `bmod`.

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobIdStr (%s)`

Job ID

`options (%d)`

Bit flags for job modification options processing

`options2 (%d)`

Bit flags for job modification options processing

`delOptions (%d)`

Delete options for the options field

`delOptions2 (%d)`

Delete options for the options2 field

`userId (%d)`

UNIX user ID of the submitter

`userName (%s)`

User name

`submitTime (%d)`

Job submission time

`umask (%d)`

File creation mask for this job

`numProcessors (%d)`

Number of processors requested for execution. The value 2147483646 means the number of processors is undefined.

`beginTime (%d)`

Start time – the job should be started on or after this time

`termTime (%d)`

Termination deadline – the job should be terminated by this time

`sigValue (%d)`

Signal value

`restartPid (%d)`

Restart process ID for the original job

`jobName (%s)`

Job name

`queue (%s)`

Name of job queue to which the job was submitted

`numAskedHosts (%d)`

Number of candidate host names

`askedHosts (%s)`

List of names of candidate hosts for job dispatching; blank if the last field value is 0. If there is more than one host name, then each additional host name will be returned in its own field

`resReq (%s)`

Resource requirements

`rLimits`

Soft CPU time limit (%d), see `getrlimit`(2)

`rLimits`

Soft file size limit (%d), see `getrlimit`(2)

`rLimits`

Soft data segment size limit (%d), see `getrlimit`2)

`rLimits`

Soft stack segment size limit (%d), see `getrlimit`(2) `rLimits`

Soft core file size limit (%d), see `getrlimit(2)`

`rLimits`

Soft memory size limit (%d), see `getrlimit(2)`

`rLimits`

Reserved (%d)

`rLimits`

Reserved (%d)

`rLimits`

Reserved (%d)

`rLimits`

Soft run time limit (%d), see `getrlimit`(2)

`rLimits`

Reserved (%d)

`hostSpec (%s)`

Model or host name for normalizing CPU time and run time

`dependCond (%s)`

Job dependency condition

`timeEvent (%d)`

Time Event, for job dependency condition; specifies when time event ended

`subHomeDir (%s)`

Submitter's home directory

`inFile (%s)`

Input file name

`outFile (%s)`

Output file name

`errFile (%s)`

Error output file name

`command (%s)`

Job command

`inFileSpool (%s)`

Spool input file

`commandSpool (%s)`

Spool command file

`chkpntPeriod (%d)`

Checkpointing period

`chkpntDir (%s)`

Checkpoint directory

`nxf (%d)`

Number of files to transfer

`xf (%s)`

List of file transfer specifications

`jobFile (%s)`

Job file name

`fromHost (%s)`

Submission host name

`cwd (%s)`

Current working directory

`preExecCmd (%s)`

Job pre-execution command

`mailUser (%s)`

Mail user name

`projectName (%s)`

Project name

`niosPort (%d)`

Callback port if batch interactive job

`maxNumProcessors (%d)`

Maximum number of processors. The value 2147483646 means the maximum number of processors is undefined.

`loginShell (%s)`

Login shell

`schedHostType (%s)`

Execution host type

`userGroup (%s)`

User group

`exceptList (%s)`

Exception handlers for the job

`userPriority (%d)`

User priority

`rsvId %s`

Advance reservation ID; for example, "`user2#0`" `jobGroup (%s)`

The job group to which the job is attached

`sla (%s)`

SLA service class name that the job is to be attached to

`extsched (%s)`

External scheduling options

`warningAction (%s)`

Job warning action

`warningTimePeriod (%d)`

Job warning time period in seconds

`licenseProject (%s)`

LSF License Scheduler project name

JOB_SIGNAL

This is created when a job is signaled with `bkill` or deleted with `bdel`. The fields are in the order they appended:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`userId (%d)`

UNIX user ID of the user invoking the command

`runCount (%d)`

Number of runs

`signalSymbol (%s)`

Signal name

`idx (%d)`

Job array index

`userName (%s)`

Name of the job submitter

JOB_EXECUTE

This is created when a job is actually running on an execution host. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`execUid (%d)`

Mapped UNIX user ID on execution host

`jobPGid (%d)`

Job process group ID

`execCwd (%s)`

Current working directory job used on execution host

`execHome (%s)`

Home directory job used on execution host

`execUsername (%s)`

Mapped user name on execution host

`jobPid (%d)`

Job process ID

`idx (%d)`

Job array index

`additionalInfo (%s)`

Placement information of HPC jobs

`SLAscaledRunLimit (%d)`

Run time limit for the job scaled by the execution host

JOB_REQUEUE

This is created when a job ended and requeued by `mbatchd`. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`idx (%d)`

Job array index

JOB_CLEAN

This is created when a job is removed from the `mbatchd` memory. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`idx (%d)`

Job array index

JOB_EXCEPTION

This is created when an exception condition is detected for a job. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`exceptMask (%d)`

Exception Id

0x01: missched

0x02: overrun

0x04: underrun

0x08: abend

0x10: cantrun

0x20: hostfail

0x40: startfail

`actMask (%d)`

Action Id

0x01: kill

0x02: alarm

0x04: rerun

0x08: setexcept

`timeEvent (%d)`

Time Event, for `missched` exception specifies when time event ended.

`exceptInfo (%d)`

Except Info, pending reason for `missched` or `cantrun` exception, the exit code of the job for the `abend` exception, otherwise 0.

`idx (%d)`

Job array index

JOB_EXT_MSG

An external message has been sent to a job. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`idx (%d)`

Job array index

`msgIdx (%d)`

Index in the list

`userId (%d)`

Unique user ID of the user invoking the command

`dataSize (%ld)`

Size of the data if it has any, otherwise 0

`postTime (%ld)`

Message sending time

`dataStatus (%d)`

Status of the attached data

`desc (%s)`

Text description of the message

`userName (%s)`

Name of the author of the message

JOB_ATTA_DATA

An update on the data status of a message for a job has been sent. The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`idx (%d)`

Job array index

`msgIdx (%d)`

Index in the list

`dataSize (%ld)`

Size of the data if is has any, otherwise 0

`dataStatus (%d)`

Status of the attached data

`fileName (%s)`

File name of the attached data

JOB_CHUNK

This is created when a job is inserted into a chunk.

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, older daemons and commands (pre-LSF Version 6.0) cannot recognize the `lsb.events` file format.

The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`membSize (%ld)`

Size of array membJobId

`membJobId (%ld)`

Job IDs of jobs in the chunk

`numExHosts (%ld)`

Number of execution hosts

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of .hosts listed in the `execHosts` field. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`execHosts (%s)`

Execution host name array

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

SBD_UNREPORTED_STATUS

This is created when an unreported status change occurs. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

actPid (%d)

Acting processing ID

jobPid (%d)

Job process ID

jobPGid (%d)

Job process group ID

newStatus (%d)

New status of the job

reason (%d)

Pending or suspending reason code, see <lsbatch/lsbatch.h>

suspreason (%d)

Pending or suspending subreason code, see <lsbatch/lsbatch.h>

lsfRusage

The following fields contain resource usage information for the job (see getrusage(2)). If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

ru_utime (%f)

User time used

ru_stime (%f)

System time used

ru_maxrss (%f)

Maximum shared text size

ru_ixrss (%f)

Integral of the shared text size over time (in KB seconds)

ru_ismrss (%f)

Integral of the shared memory size over time (valid only on Ultrix)

ru_idrss (%f)

Integral of the unshared data size over time

ru_isrss (%f)

Integral of the unshared stack size over time `ru_minflt (%f)`

Number of page reclaims

`ru_majflt (%f)`

Number of page faults

`ru_nswap (%f)`

Number of times the process was swapped out

`ru_inblock (%f)`

Number of block input operations

`ru_oublock (%f)`

Number of block output operations

`ru_ioch (%f)`

Number of characters read and written (valid only on HP-UX)

`ru_msgsnd (%f)`

Number of System V IPC messages sent

`ru_msgrcv (%f)`

Number of messages received

`ru_nsignals (%f)`

Number of signals received

`ru_nvcsw (%f)`

Number of voluntary context switches

`ru_nivcsw (%f)`

Number of involuntary context switches

`ru_exutime (%f)`

Exact user time used (valid only on ConvexOS)

`exitStatus (%d)`

Exit status of the job, see `<lsbatch/lsbatch.h>`

`execCwd (%s)`

Current working directory job used on execution host

`execHome (%s)`

Home directory job used on execution host

`execUsername (%s)`

Mapped user name on execution host

`msgId (%d)`

ID of the message

`actStatus (%d)`

Action status

1: Action started

2: One action preempted other actions

3: Action succeeded

4: Action Failed

`sigValue (%d)`

Signal value

`seq (%d)`

Sequence status of the job

`idx (%d)`

Job array index

`jRusage`

The following fields contain resource usage information for the job. If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

`mem (%d)`

Total resident memory usage in KB of all currently running processes in a given process group

`swap (%d)`

Totaly virtual memory usage in KB of all currently running processes in given process groups

`utime (%d)`

Cumulative total user time in seconds

`stime (%d)`

Cumulative total system time in seconds

`npids (%d)`

Number of currently active process in given process groups. This entry has four sub-fields:

`pid (%d)`

Process ID of the child `sbatchd` that initiated the action

`ppid (%d)`

Parent process ID

`pgid (%d)`

Process group ID

`jobId (%d)`

Process Job ID

`npgids (%d)`

Number of currently active process groups `exitInfo (%d)`

Job termination reason, see `<lsbatch/lsbatch.h>`

PRE_EXEC_START

A pre-execution command has been started.

The fields in order of occurrence are:

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`jStatus (%d)`

Job status, (4, indicating the RUN status of the job)

`jobPid (%d)`

Job process ID

`jobPGid (%d)`

Job process group ID

`hostFactor (%f)`

CPU factor of the first execution host

`numExHosts (%d)`

Number of processors used for execution

`execHosts (%s)`

List of execution host names

`queuePreCmd (%s)`

Pre-execution command

`queuePostCmd (%s)`

Post-execution command

`jFlags (%d)`

Job processing flags

`userGroup (%s)`

User group name

`idx (%d)`

Job array index

`additionalInfo (%s)`

Placement information of HPC jobs

JOB_FORCE

A job has been forced to run with `brun`.

`Version number (%s)`

The version number

`Event time (%d)`

The time of the event

`jobId (%d)`

Job ID

`userId (%d)`

UNIX user ID of the user invoking the command

`idx (%d)`

Job array index

`options (%d)`

Bit flags for job processing

`numExecHosts (%ld)`

Number of execution hosts

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of .hosts listed in the `execHosts` field. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`execHosts (%s)`

Execution host name array

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format. See "LSF_HPC_EXTENSIONS" on page 544 in "lsf.conf" for examples.

`userName (%s)`

Name of the user

`queue (%s)`

Name of job queue to which the job was submitted

SEE ALSO

Related Topics: `lsid`(1), `getrlimit`(2), `lsb_geteventrec`(3), `lsb.acct`(5), `lsb.queues`(5), `lsb.hosts`(5), `lsb.users`(5), `lsb.params`(5), `lsf.conf`(5), `lsf.cluster`(5), `badmin`(8), `bhist`(1), `mbatchd`(8)

Files: `LSB_SHAREDIR/`*`cluster_name`*`/logdir/lsb.events[.`*`n`*`]`

SEE ALSO

lsb.hosts

The `lsb.hosts` file contains host-related configuration information for the server hosts in the cluster. It is also used to define host groups and host partitions.

This file is optional. All sections are optional.

By default, this file is installed in `LSB_CONFDIR/`*`cluster_name`*`/configdir`.

Changing lsb.hosts configuration

After making any changes to `lsb.hosts`, run `badmin reconfig` to reconfigure `mbatchd`.

Contents

- "Host Section" on page 354
- "HostGroup Section" on page 358
- "HostPartition Section" on page 362
- "Automatic Time-based Configuration" on page 365

Host Section

Description

Optional. Defines the hosts, host types, and host models used as server hosts, and contains per-host configuration information. If this section is not configured, LSF uses all hosts in the cluster (the hosts listed in `lsf.cluster.`*cluster_name*) as server hosts.

Each host, host model or host type can be configured to:

- Limit the maximum number of jobs run in total
- Limit the maximum number of jobs run by each user
- Run jobs only under specific load conditions
- Run jobs only under specific time windows

The entries in a line for a host override the entries in a line for its model or type.

When you modify the cluster by adding or removing hosts, no changes are made to `lsb.hosts`. This does not affect the default configuration, but if hosts, host models, or host types are specified in this file, you should check this file whenever you make changes to the cluster and update it manually if necessary.

Host Section Structure

The first line consists of keywords identifying the load indices that you wish to configure on a per-host basis. The keyword HOST_NAME must be used; the others are optional. Load indices not listed on the keyword line do not affect scheduling decisions.

Each subsequent line describes the configuration information for one host, host model or host type. Each line must contain one entry for each keyword. Use empty parentheses ( ) or a dash (-) to specify the default value for an entry.

HOST_NAME

Required. Specify the name, model, or type of a host, or the keyword `default`.

host name The name of a host defined in `lsf.cluster.`*cluster_name*. The official host name returned by `gethostbyname(3)`.

host model A host model defined in `lsf.shared`.

host type A host type defined in `lsf.shared`.

default The reserved host name `default` indicates all hosts in the cluster not otherwise referenced in the section (by name or by listing its model or type).

CHKPNT

Description If C, checkpoint copy is enabled. With checkpoint copy, all opened files are automatically copied to the checkpoint directory by the operating system when a process is checkpointed.

Example

```
HOST_NAME   CHKPNT
hostA          C
```

Compatibility Checkpoint copy is only supported on Cray systems.

Default No checkpoint copy.

DISPATCH_WINDOW

Description The time windows in which jobs from this host, host model, or host type are dispatched. Once dispatched, jobs are no longer affected by the dispatch window.

Default Undefined (always open).

EXIT_RATE

Description Specifies a threshold in minutes for exited jobs. If the job exit rate is exceeded for 10 minutes or the period specified by JOB_EXIT_RATE_DURATION, LSF invokes `LSF_SERVERDIR/eadmin` to trigger a host exception.

Example The following Host section defines a job exit rate of 20 jobs per minute for all hosts:

```
Begin Host
HOST_NAME    MXJ      EXIT_RATE
Default      !        20
End Host
```

Default Undefined

JL/U

Description Per-user job slot limit for the host. Maximum number of job slots that each user can use on this host.

Example

```
HOST_NAME  JL/U
hostA         2
```

Default Unlimited

MIG

Description Enables job migration and specifies the migration threshold, in minutes.

If a checkpointable or rerunnable job dispatched to the host is suspended (SSUSP state) for longer than the specified number of minutes, the job is migrated. A value of 0 specifies that a suspended job should be migrated immediately.

If a migration threshold is defined at both host and queue levels, the lower threshold is used.

If you do not want migrating jobs to be run or restarted immediately, set LSB_MIG2PEND in `lsf.conf` so that migrating jobs are considered as pending jobs and inserted in the pending jobs queue.

If you want migrating jobs to be considered as pending jobs but you want them to be placed at the bottom of the queue without considering submission time, define both LSB_MIG2PEND and LSB_REQUEUE_TO_BOTTOM in `lsf.conf`.

Example

```
HOST_NAME   MIG
hostA        10
```

In this example, the migration threshold is 10 minutes.

Default Undefined (no migration)

MXJ

Description The number of job slots on the host.

With MultiCluster resource leasing model, this is the number of job slots on the host that are available to the local cluster.

Use “!” to make the number of job slots equal to the number of CPUs on a host.

For the reserved host name `default`, “!” makes the number of job slots equal to the number of CPUs on all hosts in the cluster not otherwise referenced in the section.

By default, the number of running and suspended jobs on a host cannot exceed the number of job slots. If preemptive scheduling is used, the suspended jobs are not counted as using a job slot.

On multiprocessor hosts, to fully use the CPU resource, make the number of job slots equal to or greater than the number of processors.

Default Unlimited

load_index

Syntax *load_index*

loadSched[**/***loadStop*]

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index as a column. Specify multiple columns to configure thresholds for multiple load indices.

Description Scheduling and suspending thresholds for dynamic load indices supported by LIM, including external load indices.

Each load index column must contain either the default entry or two numbers separated by a slash ‘/’, with no white space. The first number is the scheduling threshold for the load index; the second number is the suspending threshold.

Queue-level scheduling and suspending thresholds are defined in `lsb.queues`. If both files specify thresholds for an index, those that apply are the most restrictive ones.

Example

```
HOST_NAME    mem     swp
hostA        100/10  200/30
```

This example translates into a `loadSched` condition of

```
mem>=100 && swp>=200
```

and a `loadStop` condition of

```
mem < 10 || swp < 30
```

Default Undefined

Example of a Host Section

```
Begin Host
HOST_NAME MXJ JL/U r1m     pg    DISPATCH_WINDOW
hostA     1   -    0.6/1.6 10/20 (5:19:00-1:8:30 20:00-8:30)
SUNSOL    1   -    0.5/2.5 -     23:00-8:00
default   2   1    0.6/1.6 20/40 ()
End Host
```

SUNSOL is a host type defined in `lsf.shared`. This example `Host` section configures one host and one host type explicitly and configures default values for all other load-sharing hosts.

`HostA` runs one batch job at a time. A job will only be started on `hostA` if the `r1m` index is below 0.6 and the `pg` index is below 10; the running job is stopped if the `r1m` index goes above 1.6 or the `pg` index goes above 20. `HostA` only accepts batch jobs from 19:00 on Friday evening until 8:30 Monday morning and overnight from 20:00 to 8:30 on all other days.

For hosts of type SUNSOL, the `pg` index does not have host-specific thresholds and such hosts are only available overnight from 23:00 to 8:00.

The entry with host name default applies to each of the other hosts in the cluster. Each host can run up to two jobs at the same time, with at most one job from each user. These hosts are available to run jobs at all times. Jobs may be started if the `r1m` index is below 0.6 and the `pg` index is below 20, and a job from the lowest priority queue is suspended if `r1m` goes above 1.6 or `pg` goes above 40.

HostGroup Section

Description

Optional. Defines host groups.

The name of the host group can then be used in other host group, host partition, and queue definitions, as well as on the command line. Specifying the name of a host group has exactly the same effect as listing the names of all the hosts in the group.

Structure

Host groups are specified in the same format as user groups in `lsb.users`.

The first line consists of two mandatory keywords, GROUP_NAME and GROUP_MEMBER, and an optional keyword, CONDENSE. Subsequent lines name a group and list its membership.

The sum of host groups and host partitions cannot be more than MAX_GROUPS (see `lsbatch.h` for details).

GROUP_NAME

Description

An alphanumeric string representing the name of the host group.

You cannot use the reserved name `all`, and group names must not conflict with host names.

CONDENSE

Description

Optional. Defines condensed host groups.

Condensed host groups are displayed in a condensed output format for the `bhosts` and `bjobs` commands.

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.

Valid Values

`Y` or `N`.

Default

`N`. The specified host group is not condensed.

GROUP_MEMBER

Description

A space-delimited list of host names or previously defined host group names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of hosts and host groups can appear on multiple lines because hosts can belong to multiple groups. The reserved name `all` specifies all hosts in the cluster. Use an exclamation mark (`!`) to specify that the group membership should be retrieved via `egroup`.

Pattern definition

You can use string literals and special characters when defining host group members. Each entry cannot contain any spaces, as the list itself is space delimited.

When a leased-in host joins the cluster, the host name is in the form of *host@cluster*. For these hosts, only the host part of the host name is subject to pattern definitions.

You can use the following special characters to specify host group members:

- Use a tilde (~) to exclude specified hosts or host groups from the list.
- Use an asterisk (*) as a wildcard character to represent any number of characters.
- Use square brackets with a hyphen ([*integer1* - *integer2*]) to define a range of non-negative integers at the end of a host name. The first integer must be less than the second integer.
- Use square brackets with commas ([*integer1* , *integer2* ...]) to define individual non-negative integers at the end of a host name.
- Use square brackets with commas and hyphens (for example, [*integer1* - *integer2*, *integer3* , *integer4* - *integer5*]) to define different ranges of non-negative integers at the end of a host name.

Restrictions

- You cannot use more than one set of square brackets in a single host group definition.

  The following example is *not* correct:

  ```
  ... (hostA[1-10]B[1-20] hostC[101-120])
  ```

  The following example is correct:

  ```
  ... (hostA[1-20] hostC[101-120])
  ```

- You cannot define subgroups that contain wildcards and special characters. The following definition for `groupB` is not correct because `groupA` defines hosts with a wildcard:

  ```
  Begin HostGroup
  GROUP_NAME    GROUP_MEMBER
  groupA        (hostA*)
  groupB        (groupA)
  End HostGroup
  ```

Example HostGroup Sections

Example 1

```
Begin HostGroup
GROUP_NAME   GROUP_MEMBER
groupA       (hostA hostD)
groupB       (hostF groupA hostK)
groupC       (!)
End HostGroup
```

This example defines three host groups:

- `groupA` includes `hostA` and `hostD`.
- `groupB` includes `hostF` and `hostK`, along with all hosts in `groupA`.
- The group membership of `groupC` will be retrieved via `egroup`.

Example 2

```
Begin HostGroup
GROUP_NAME    GROUP_MEMBER
groupA        (all)
groupB        (groupA ~hostA ~hostB)
groupC        (hostX hostY hostZ)
groupD        (groupC ~hostX)
groupE        (all ~groupC ~hostB)
groupF        (hostF groupC hostK)
End HostGroup
```

This example defines the following host groups:

- `groupA` contains all hosts in the cluster.
- `groupB` contains all the hosts in the cluster except for `hostA` and `hostB`.
- `groupC` contains only `hostX`, `hostY`, and `hostZ`.
- `groupD` contains the hosts in `groupC` except for `hostX`. Note that `hostX` must be a member of host group `groupC` to be excluded from `groupD`.
- `groupE` contains all hosts in the cluster excluding the hosts in `groupC` and `hostB`.
- `groupF` contains `hostF`, `hostK`, and the 3 hosts in `groupC`.

Example 3

```
Begin HostGroup
GROUP_NAME    CONDENSE    GROUP_MEMBER
groupA           N        (all)
groupB           N        (hostA, hostB)
groupC           Y        (all)
```

This example defines the following host groups:

- `groupA` shows uncondensed output and contains all hosts in the cluster.
- `groupB` shows uncondensed output, and contains `hostA` and `hostB`.
- `groupC` shows condensed output and contains all hosts in the cluster.

Example 4

```
Begin HostGroup
GROUP_NAME    CONDENSE    GROUP_MEMBER
groupA           Y        (host*)
groupB           N        (*A)
groupC           N        (hostB* ~hostB[1-50])
groupD           Y        (hostC[1-50] hostC[101-150])
groupE           N        (hostC[51-100] hostC[151-200])
groupF           Y        (hostD[1,3] hostD[5-10])
groupG           N        (hostD[11-50] ~hostD[15,20,25] hostD2)
End HostGroup
```

This example defines the following host groups:

- `groupA` shows condensed output, and contains all hosts starting with the string `host`.
- `groupB` shows uncondensed output, and contains all hosts ending with the string `A`, such as `hostA`.
- `groupC` shows uncondensed output, and contains all hosts starting with the string `hostB` except for the hosts from `hostB1` to `hostB50`.
- `groupD` shows condensed ouput, and contains all hosts from `hostC1` to `hostC50` and all hosts from `hostC101` to `hostC150`.

- `groupE` shows uncondensed output, and contains all hosts from `hostC51` to `hostC100` and all hosts from `hostC151` to `hostC200`.
- `groupF` shows condensed output, and contains `hostD1`, `hostD3`, and all hosts from `hostD5` to `hostD10`.
- `groupG` shows uncondensed output, and contains all hosts from `hostD11` to `hostD50` except for `hostD15`, `hostD20`, and `hostD25`. `groupG` also includes `hostD2`.

HostPartition Section

Description

Optional; used with host partition user-based fairshare scheduling. Defines a host partition, which defines a user-based fairshare policy at the host level.

Configure multiple sections to define multiple partitions.

The members of a host partition form a host group with the same name as the host partition.

Limitations on Queue Configuration

- If you configure a host partition, you cannot configure fairshare at the queue level.
- If a queue uses a host that belongs to a host partition, it should not use any hosts that don't belong to that partition. All the hosts in the queue should belong to the same partition. Otherwise, you might notice unpredictable scheduling behavior:
  - Jobs in the queue sometimes may be dispatched to the host partition even though hosts not belonging to any host partition have a lighter load.
  - If some hosts belong to one host partition and some hosts belong to another, only the priorities of one host partition are used when dispatching a parallel job to hosts from more than one host partition.

Shared Resources and Host Partitions

- If a resource is shared among hosts included in host partitions and hosts that are not included in any host partition, jobs in queues that use the host partitions will always get the shared resource first, regardless of queue priority.
- If a resource is shared among host partitions, jobs in queues that use the host partitions listed first in the `HostPartition` section of `lsb.hosts` will always have priority to get the shared resource first. To allocate shared resources among host partitions, LSF considers host partitions in the order they are listed in `lsb.hosts`.

Structure

Each host partition always consists of 3 lines, defining the name of the partition, the hosts included in the partition, and the user share assignments.

HPART_NAME

Syntax

HPART_NAME=*partition_name*

Description

Specifies the name of the partition.

HOSTS

Syntax

HOSTS=[[**~**]*host_name* | [**~**]*host_group* | **all**]...

Description

Specifies the hosts in the partition, in a space-separated list.

A host cannot belong to multiple partitions.

A host group cannot be empty.

Hosts that are not included in any host partition are controlled by the FCFS scheduling policy instead of the fairshare scheduling policy.

Optionally, use the reserved host name `all` to configure a single partition that applies to all hosts in a cluster.

Optionally, use the not operator (~) to exclude hosts or host groups from the list of hosts in the host partition.

Examples

`HOSTS=all ~hostK ~hostM`

The partition includes all the hosts in the cluster, except for `hostK` and `hostM`.

`HOSTS=groupA ~hostL`

The partition includes all the hosts in host group `groupA` except for `hostL`.

USER_SHARES

Syntax

USER_SHARES=[*user***,** *number_shares***]** ...

Description

Specifies user share assignments

- Specify at least one user share assignment.
- Enclose each user share assignment in square brackets, as shown.
- Separate a list of multiple share assignments with a space between the square brackets.
- *user*

  Specify users who are also configured to use the host partition. You can assign the shares:

  - To a single user (specify *user_name*)
  - To users in a group, individually (specify *group_name***@**) or collectively (specify *group_name*)
  - To users not included in any other share assignment, individually (specify the keyword `default`) or collectively (specify the keyword `others`)

  By default, when resources are assigned collectively to a group, the group members compete for the resources according to FCFS scheduling. You can use hierarchical fairshare to further divide the shares among the group members.

  When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.
- *number_shares*

  Specify a positive integer representing the number of shares of the cluster resources assigned to the user.

  The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

Example of a HostPartition Section

```
Begin HostPartition
HPART_NAME = Partition1
HOSTS = hostA hostB
USER_SHARES = [groupA@, 3] [groupB, 7] [default, 1]
End HostPartition
```

Automatic Time-based Configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.hosts` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the `badmin reconfig` command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

In the following example, the `#if`, `#else`, `#endif` are not interpreted as comments by LSF but as if-else constructs.

```
Begin Host
HOST_NAME   r15s   r1m   pg
host1       3/5    3/5   12/20
#if time(5:16:30-1:8:30 20:00-8:30)
host2       3/5    3/5   12/20
#else0host2       2/3    2/3   10/12
#endif
host3       3/5    3/5   12/20
End Host
```

lsb.modules

The lsb.modules file contains configuration information for LSF scheduler and resource broker modules. The file contains only one section, named PluginModule.

This file is optional. If no scheduler or resource broker modules are configured, LSF uses the default scheduler plugin modules named schmod_default and schmod_fcfs.

The lsb.modules file is stored in the directory LSB_CONFDIR/*cluster_name*/configdir, where LSB_CONFDIR is defined in lsf.conf.

Changing lsb.modules configuration

After making any changes to lsb.modules, run badmin reconfig to reconfigure mbatchd.

Contents

- "PluginModule Section" on page 368

PluginModule Section

Description

Defines the plugin modules for the LSF scheduler and LSF resource broker. If this section is not configured, LSF uses the default scheduler plugin modules named `schmod_default` and `schmod_fcfs`, which enable the LSF default scheduling features.

Example PluginModule section

The following PluginModule section enables all scheduling policies provided by LSF:

```
Begin PluginModule
SCH_PLUGIN              RB_PLUGIN          SCH_DISABLE_PHASES
schmod_default             ()                     ()
schmod_fairshare           ()                     ()
schmod_fcfs                ()                     ()
schmod_limit               ()                     ()
schmod_parallel            ()                     ()
schmod_reserve             ()                     ()
schmod_preemption          ()                     ()
schmod_advrsv              ()                     ()
schmod_mc                  ()                     ()
schmod_cpuset              ()                     ()
End PluginModule
```

PluginModule section structure

The first line consists of the following keywords:

- SCH_PLUGIN
- RB_PLUGIN
- SCH_DISABLE_PHASES

They identify the scheduler plugins, resource broker plugins, and the scheduler phase to be disabled for the plugins that you wish to configure.

Each subsequent line describes the configuration information for one scheduler plugin module, resource broker plugin module, and scheduler phase, if any, to be disabled for the plugin. Each line must contain one entry for each keyword. Use empty parentheses ( ) or a dash (-) to specify the default value for an entry.

SCH_PLUGIN

Description

Required. The SCH_PLUGIN column specifies the shared module name for the LSF scheduler plugin. Each plugin requires a corresponding license. Scheduler plugins are called in the order they are listed in the PluginModule section.

By default, all shared modules for scheduler plugins are located in LSF_LIBDIR. On UNIX, you can also specify a full path to the name of the scheduler plugin.

The following modules are supplied with LSF:

schmod_default

Enables the default LSF scheduler features.

Licensed by: `LSF_Manager`

schmod_fcfs

Enables the first-come, first-served (FCFS) scheduler features. `schmod_fcfs` can appear anywhere in the SCH_PLUGIN list. By default, if `schmod_fcfs` is not configured in `lsb.modules`, it is loaded automatically along with `schmod_default`.

Source code (`sch.mod.fcfs.c`) for the `schmod_fcfs` scheduler plugin module is installed in the directory

`LSF_TOP/6.2/misc/examples/external_plugin/`

Use the LSF scheduler plugin SDK to modify the FCFS scheduler module code to suit the job scheduling requirements of your site.

See *Using the Platform LSF SDK* for more detailed information about writing, building, and configuring your own custom scheduler plugins.

schmod_fairshare

Enables the LSF fairshare scheduling features.

Licensed by: `LSF_Sched_Fairshare`

schmod_limit

Enables the LSF resource allocation limit features.

Licensed by: `LSF_Manager`

schmod_parallel

Enables scheduling of parallel jobs submitted with `bsub -n`.

Licensed by: `LSF_Sched_Parallel`

schmod_reserve

Enables the LSF resource reservation features.

To enable processor reservation, backfill, and memory reservation for parallel jobs, you must configure both `schmod_parallel` and `schmod_reserve` in `lsb.modules`. If only `schmod_reserve` is configured, backfill and memory reservation are enabled only for sequential jobs, and processor reservation is not enabled.

Licensed by: `LSF_Sched_Resource_Reservation`

schmod_preemption

Enables the following LSF preemption scheduler features.

Licensed by: `LSF_Sched_Preemption`

schmod_advrsv

Handles jobs that use advance reservations (`brsvadd`, `brsvs`, `brsvdel`, `bsub -U`)

Licensed by: `LSF_Sched_Advance_Reservation`

schmod_topology

Obsolete. Replaced by `schmod_cpuset`.

schmod_cpuset

Handles jobs that use IRIX cpusets
(`bsub -extsched "CPUSET[`*`cpuset_options`*`]"`)

The `schmod_cpuset` plugin name must be the last plugin the PluginModule list.

schmod_mc

Enables MultiCluster job forwarding

Licensed by: `LSF_MultiCluster`

Scheduler plugin SDK

Use the LSF scheduler plugin SDK to write customized scheduler modules that give you more flexibility and control over job scheduling. Enable your custom scheduling policies by configuring your modules under SCH_PLUGIN in the PluginModules section of `lsb.modules`.

The directory

`LSF_TOP/6.2/misc/examples/external_plugin/`

contains sample plugin code. See *Using the Platform LSF SDK* for more detailed information about writing, building, and configuring your own custom scheduler plugins.

schmod_jobweight

An optional scheduler plugin module to enable Cross-Queue Job Weight scheduling policies. The `schmod_jobweight` plugin must be listed before `schmod_cpuset` and `schmod_rms`, and after all other scheduler plugin modules.

You should not use job weight scheduling together with fairshare scheduling or job preemption. To avoid scheduling conflicts, you should comment out `schmod_fairshare` and `schmod_preemption` in `lsb.modules`.

RB_PLUGIN

Description

RB_PLUGIN specifies the shared module name for resource broker plugins. Resource broker plugins collect and update job resource accounting information, and provide it to the scheduler.

Normally, for each scheduler plugin module, there is a corresponding resource broker plugin module to support it. However, the resource broker also supports multiple plugin modules for one scheduler plugin module.

For example, a fairshare policy may need more than one resource broker plugin module to support it if the policy has multiple configurations.

A scheduler plugin can have one, multiple, or none RB plugins corresponding to it.

Example

```
NAME                    RB_PLUGIN
schmod_default         ()
schmod_fairshare       (rb_fairshare)
```

Default

Undefined

SCH_DISABLE_PHASES

Description

SCH_DISABLE_PHASES specifies which scheduler phases, if any, to be disabled for the plugin. LSF scheduling has four phases:

1. Preprocessing—the scheduler checks the readiness of the job for scheduling and prepares a list of ready resource seekers. It also checks the start time of a job, and evaluates any job dependencies.
2. Match/limit—the scheduler evaluates the job resource requirements and prepares candidate hosts for jobs by matching jobs with resources. It also applies resource allocation limits. Jobs with all required resources matched go on to order/allocation phase.

   Not all jobs are mapped to all potential available resources. Jobs without any matching resources will not go through the Order/Allocation Phase but can go through the Post-processing phase, where preemption may be applied to get resources the job needs to run.
3. Order/allocation—the scheduler sorts jobs with matched resources and allocates resources for each job, assigning job slot, memory, and other resources to the job. It also checks if the allocation satisfies all constraints defined in configuration, such as queue slot limit, deadline for the job, etc.

In the order phase, the scheduler applies policies such as FCFS, Fairshare and Host-partition and consider job priorities within user groups and share groups. By default, job priority within a pool of jobs from the same user is based on how long the job has been pending.

For resource intensive jobs (jobs requiring a lot of CPUs or a large amount of memory), resource reservation is performed so that these jobs are not starved.

When all the currently available resources are allocated, jobs go on to post-processing.

4 Post-processing—the scheduler prepares jobs from the order/allocation phase for dispatch and applies preemption or backfill policies to obtain resources for the jobs that have completed pre-processing or match/limit phases, but did not have resources available to enter the next scheduling phase.

Each scheduler plugin module invokes one or more scheduler phase. The processing for a give phase can be disabled or skipped if:

The plugin module does not need to do any processing for that phase or the processing has already been done by a previous plugin module in the list.

The scheduler will not invoke phases marked by SCH_DISABLE_PHASES when scheduling jobs.

None of the plugins provided by LSF should require phases to be disabled, but your own custom plugin modules using the scheduler SDK may need to disable one or more scheduler phases.

Example In the following configuration, the `schmod_custom` plugin module disables the order allocation (3) and post-processing (4) phases:

```
NAME               SCH_DISABLE_PHASES
schmod_default     ()
schmod_custom      (3,4)
```

Default Undefined

SEE ALSO

`lsf.cluster`(5), `lsf.conf`(5), `mbschd`(8)

lsb.params

The lsb.params file defines general parameters used by the LSF system. This file contains only one section, named Parameters. mbatchd uses lsb.params for initialization. The file is optional. If not present, the LSF-defined defaults are assumed.

Some of the parameters that can be defined in lsb.params control timing within the system. The default settings provide good throughput for long-running batch jobs while adding a minimum of processing overhead in the batch daemons.

This file is installed by default in LSB_CONFDIR/*cluster_name*/configdir.

Changing lsb.params configuration

After making any changes to lsb.params, run badmin reconfig to reconfigure mbatchd.

Contents

- "Parameters Section" on page 374
- "Automatic Time-based Configuration" on page 396

Parameters Section

This section and all the keywords in this section are optional. If keywords are not present, the default values are assumed.

Parameters

- ABS_RUNLIMIT
- ACCT_ARCHIVE_AGE
- ACCT_ARCHIVE_SIZE
- ACCT_ARCHIVE_TIME
- CHUNK_JOB_DURATION
- CLEAN_PERIOD
- CONDENSE_PENDING_REASONS
- CPU_TIME_FACTOR
- COMMITTED_RUN_TIME_FACTOR
- DEFAULT_HOST_SPEC
- DEFAULT_PROJECT
- DEFAULT_QUEUE
- DETECT_IDLE_JOB_AFTER
- DISABLE_UACCT_MAP
- EADMIN_TRIGGER_DURATION
- ENABLE_HIST_RUN_TIME
- ENABLE_USER_RESUME
- EVENT_UPDATE_INTERVAL
- HIST_HOURS
- JOB_ACCEPT_INTERVAL
- JOB_ATTA_DIR
- JOB_DEP_LAST_SUB
- JOB_EXIT_RATE_DURATION
- JOB_POSITION_CONTROL_BY_ADMIN
- JOB_PRIORITY_OVER_TIME
- JOB_SCHEDULING_INTERVAL
- JOB_SPOOL_DIR
- JOB_TERMINATE_INTERVAL
- MAX_ACCT_ARCHIVE_FILE
- MAX_CONCURRENT_JOB_QUERY
- MAX_INFO_DIRS
- MAX_JOB_ARRAY_SIZE
- MAX_JOB_ATTA_SIZE
- MAX_JOBID
- MAX_JOBINFO_QUERY_PERIOD
- MAX_JOB_MSG_NUM
- MAX_JOB_NUM

- MAX_PEND_JOBS
- MAX_PREEXEC_RETRY
- MAX_SBD_CONNS
- MAX_SBD_FAIL
- MAX_USER_PRIORITY
- MBD_REFRESH_TIME
- MBD_SLEEP_TIME
- MC_RECLAIM_DELAY
- MC_PENDING_REASON_PKG_SIZE
- MC_PENDING_REASON_UPDATE_INTERVAL
- MC_RUSAGE_UPDATE_INTERVAL
- MIN_SWITCH_PERIOD
- NO_PREEMPT_RUN_TIME
- NO_PREEMPT_FINISH_TIME
- NQS_QUEUES_FLAGS
- NQS_REQUESTS_FLAGS
- PARALLEL_SCHED_BY_SLOT
- PEND_REASON_UPDATE_INTERVAL
- PEND_REASON_MAX_JOBS
- PG_SUSP_IT
- PREEMPTABLE_RESOURCES
- PREEMPT_FOR
- PREEMPTION_WAIT_TIME
- RESOURCE_RESERVE_PER_SLOT
- RUN_JOB_FACTOR
- RUN_TIME_FACTOR
- SBD_SLEEP_TIME
- SUB_TRY_INTERVAL
- SYSTEM_MAPPING_ACCOUNT

ABS_RUNLIMIT

Syntax **ABS_RUNLIMIT=y** | **Y**

Description If set, the run time limit specified by the `-W` option of `bsub`, or the RUNLIMIT queue parameter in `lsb.queues` is not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted with a run limit.

Default Undefined. Run limit is normalized.

ACCT_ARCHIVE_AGE

Syntax **ACCT_ARCHIVE_AGE=***days*

Description Enables automatic archiving of LSF accounting log files, and specifies the archive interval. LSF archives the current log file if the length of time from its creation date exceeds the specified number of days.

See also

- ACCT_ARCHIVE_SIZE also enables automatic archiving.
- ACCT_ARCHIVE_TIME also enables automatic archiving.
- MAX_ACCT_ARCHIVE_FILE enables automatic deletion of the archives.

Default Undefined (no limit to the age of `lsb.acct`).

ACCT_ARCHIVE_SIZE

Syntax **`ACCT_ARCHIVE_SIZE=`***kilobytes*

Description Enables automatic archiving of LSF accounting log files, and specifies the archive threshold. LSF archives the current log file if its size exceeds the specified number of kilobytes.

See also

- ACCT_ARCHIVE_AGE also enables automatic archiving.
- ACCT_ARCHIVE_TIME also enables automatic archiving.
- MAX_ACCT_ARCHIVE_FILE enables automatic deletion of the archives.

Default Undefined (no limit to the size of `lsb.acct`).

ACCT_ARCHIVE_TIME

Syntax **`ACCT_ARCHIVE_TIME=`***hh***`:`***mm*

Description Enables automatic archiving of LSF accounting log file `lsb.acct`, and specifies the time of day to archive the current log file.

See also

- ACCT_ARCHIVE_AGE also enables automatic archiving.
- ACCT_ARCHIVE_SIZE also enables automatic archiving.
- MAX_ACCT_ARCHIVE_FILE enables automatic deletion of the archives.

Default Undefined (no time set for archiving `lsb.acct`).

CHUNK_JOB_DURATION

Syntax **`CHUNK_JOB_DURATION=`***minutes*

Description Specifies a CPU limit or run limit for jobs submitted to a chunk job queue to be chunked.

When CHUNK_JOB_DURATION is set, the CPU limit or run limit set in the queue (CPULIMIT or RUNLMIT) or specified at job submission (`-c` or `-W` `bsub` options) must be less than or equal to CHUNK_JOB_DURATION for jobs to be chunked.

If CHUNK_JOB_DURATION is set, jobs are *not* chunked if:

- No CPU limit and no run limit are specified in the queue (CPULIMIT and RUNLIMIT) or at job submission (`-c` or `-W` `bsub` options).

  or
- CPU limit or a run limit is greater than the value of CHUNK_JOB_DURATION.

If CHUNK_JOB_DURATION is set, chunk jobs are accepted regardless of the value of CPULIMIT or RUNLIMIT.

The value of CHUNK_JOB_DURATION is displayed by `bparams -l`.

Examples

- CHUNK_JOB_DURATION is not defined:

    - Jobs with no CPU limit or run limit are chunked
    - Jobs with CPU limit or run limit less than or equal to 30 are chunked
    - Jobs with CPU limit or run limit greater than 30 are *not* chunked
- CHUNK_JOB_DURATION=90:
    - Jobs with no CPU limit or run limit are *not* chunked
    - Jobs with CPU limit or run limit less than or equal to 90 are chunked
    - Jobs with CPU limit or run limit greater than 90 are *not* chunked

Default Undefined

CLEAN_PERIOD

Syntax **`CLEAN_PERIOD=`***seconds*

Description For non-repetitive jobs, the amount of time that job records for jobs that have finished or have been killed are kept in `mbatchd` core memory after they have finished.

Users can still see all jobs after they have finished using the `bjobs` command.

For jobs that finished more than CLEAN_PERIOD seconds ago, use the `bhist` command.

Default 3600 (1 hour)

CONDENSE_PENDING_REASONS

Syntax **`CONDENSE_PENDING_REASONS=Y | N`**

Description If enabled, condenses all host-based pending reasons into one generic pending reason.

If enabled, you can request a full pending reason list by running the following command:

```
% badmin diagnose jobId
```

You must be LSF administrator or a queue administrator to run this command.

Examples

- `CONDENSE_PENDING_REASONS=Y`

  If a job has no other pending reason, `bjobs -p` or `bjobs -l` displays the following:

  `Individual host based reasons`
- `CONDENSE_PENDING_REASONS=N`

  The pending reasons are not suppressed. Host-based pending reasons are displayed.

Default `N`

CPU_TIME_FACTOR

Syntax **`CPU_TIME_FACTOR=`***number*

Description Used only with fairshare scheduling. CPU time weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the cumulative CPU time used by a user's jobs.

Default 0.7

COMMITTED_RUN_TIME_FACTOR

Syntax **COMMITTED_RUN_TIME_FACTOR=***number*

Description Used only with fairshare scheduling. Committed run time weighting factor.

In the calculation of a user's dynamic priority, this factor determines the relative importance of the committed run time in the calculation. If the `-W` option of `bsub` is not specified at job submission and a RUNLIMIT has not been set for the queue, the committed run time is not considered.

Valid Values Any positive number between 0.0 and 1.0

Default 0.0

DEFAULT_HOST_SPEC

Syntax **DEFAULT_HOST_SPEC=***host_name* | *host_model*

Description The default CPU time normalization host for the cluster.

The CPU factor of the specified host or host model will be used to normalize the CPU time limit of all jobs in the cluster, unless the CPU time normalization host is specified at the queue or job level.

Default Undefined

DEFAULT_PROJECT

Syntax **DEFAULT_PROJECT=***project_name*

Description The name of the default project. Specify any string.

When you submit a job without specifying any project name, and the environment variable LSB_DEFAULTPROJECT is not set, LSF automatically assigns the job to this project.

Default `default`

DEFAULT_QUEUE

Syntax **DEFAULT_QUEUE=***queue_name* ...

Description Space-separated list of candidate default queues (candidates must already be defined in `lsb.queues`).

When you submit a job to LSF without explicitly specifying a queue, and the environment variable LSB_DEFAULTQUEUE is not set, LSF puts the job in the first queue in this list that satisfies the job's specifications subject to other restrictions, such as requested hosts, queue status, etc.

Default Undefined. When a user submits a job to LSF without explicitly specifying a queue, and there are no candidate default queues defined (by this parameter or by the user's environment variable LSB_DEFAULTQUEUE), LSF automatically creates a new queue named `default`, using the default configuration, and submits the job to that queue.

DETECT_IDLE_JOB_AFTER

Syntax **DETECT_IDLE_JOB_AFTER=***time_minutes*

Description The minimum job run time before `mbatchd` reports that the job is idle.

Default 20 (`mbatchd` checks if the job is idle after 20 minutes of run time)

DISABLE_UACCT_MAP

Syntax **DISABLE_UACCT_MAP=y | Y**

Description Specify y or Y to disable user-level account mapping.

Default Undefined

EADMIN_TRIGGER_DURATION

Description Defines how often `LSF_SERVERDIR/eadmin` is invoked once a job exception is detected. Used in conjunction with job exception handling parameters JOB_OVERRUN and JOB_UNDERRUN in `lsb.queues`.

Example `EADMIN_TRIGGER_DURATION=20`

Default 5 minutes

ENABLE_HIST_RUN_TIME

Syntax **ENABLE_HIST_RUN_TIME=y | Y**

Description Used only with fairshare scheduling. If set, enables the use of historical run time in the calculation of fairshare scheduling priority.

Default Undefined

ENABLE_USER_RESUME

Syntax **ENABLE_USER_RESUME=Y | N**

Description Defines job resume permissions.

When this parameter is defined:

- If the value is Y, users can resume their own jobs that have been suspended by the administrator.
- If the value is N, jobs that are suspended by the administrator can only be resumed by the administrator or `root`; users do not have permission to resume a job suspended by another user or the administrator. Administrators can resume jobs suspended by users or administrators.

Default Undefined (users cannot resume jobs suspended by administrator)

EVENT_UPDATE_INTERVAL

Syntax **EVENT_UPDATE_INTERVAL=***seconds*

Description Used with duplicate logging of event and accounting log files. LSB_LOCALDIR in `lsf.conf` must also be specified. Specifies how often to back up the data and synchronize the directories (LSB_SHAREDIR and LSB_LOCALDIR).

The directories are always synchronized when data is logged to the files, or when `mbatchd` is started on the first LSF master host.

Use this parameter if NFS traffic is too high and you want to reduce network traffic.

Valid Values 1 to INFINIT_INT

INFINIT_INT is defined in `lsf.h`

Default Undefined

See also See "lsf.conf" under "LSB_LOCALDIR" on page 523.

HIST_HOURS

Syntax **`HIST_HOURS=`***hours*

Description Used only with fairshare scheduling. Determines a rate of decay for cumulative CPU time and historical run time.

To calculate dynamic user priority, LSF scales the actual CPU time using a decay factor, so that 1 hour of recently-used time is equivalent to 0.1 hours after the specified number of hours has elapsed.

To calculate dynamic user priority with historical run time, LSF scales the accumulated run time of finished jobs using the same decay factor, so that 1 hour of recently-used time is equivalent to 0.1 hours after the specified number of hours has elapsed.

When HIST_HOURS=0, CPU time accumulated by running jobs is not decayed.

Default 5

JOB_ACCEPT_INTERVAL

Syntax **`JOB_ACCEPT_INTERVAL=`***integer*

Description The number you specify is multiplied by the value of `lsb.params` MBD_SLEEP_TIME (60 seconds by default). The result of the calculation is the number of seconds to wait after dispatching a job to a host, before dispatching a second job to the same host.

If 0 (zero), a host may accept more than one job. By default, there is no limit to the total number of jobs that can run on a host, so if this parameter is set to 0, a very large number of jobs might be dispatched to a host all at once. This can overload your system to the point that it will be unable to create any more processes. It is not recommended to set this parameter to 0.

JOB_ACCEPT_INTERVAL set at the queue level (`lsb.queues`) overrides JOB_ACCEPT_INTERVAL set at the cluster level (`lsb.params`).

Default 1

JOB_ATTA_DIR

Syntax **`JOB_ATTA_DIR=`***directory*

Description The shared directory in which `mbatchd` saves the attached data of messages posted with the `bpost` command.

Use JOB_ATTA_DIR if you use `bpost`(1) and `bread`(1)to transfer large data files between jobs and want to avoid using space in LSB_SHAREDDIR. By default, the `bread`(1) command reads attachment data from the JOB_ATTA_DIR directory.

JOB_ATTA_DIR should be shared by all hosts in the cluster, so that any potential LSF master host can reach it. Like LSB_SHAREDIR, the directory should be owned and writable by the primary LSF administrator. The directory must have at least 1 MB of free space.

The attached data will be stored under the directory in the format:

`JOB_ATTA_DIR/`*`timestamp.jobid`*`.msgs/msg$msgindex`

On UNIX, specify an absolute path. For example:

`JOB_ATTA_DIR=/opt/share/lsf_work`

On Windows, specify a UNC path or a path with a drive letter. For example:

`JOB_ATTA_DIR=\\HostA\temp\lsf_workor`

`JOB_ATTA_DIR=D:\temp\lsf_work`

After adding JOB_ATTA_DIR to `lsb.params`, use `badmin reconfig` to reconfigure your cluster.

Valid values JOB_ATTA_DIR can be any valid UNIX or Windows path up to a maximum length of 256 characters.

Default Undefined

If JOB_ATTA_DIR is not specified, job message attachments are saved in `LSB_SHAREDIR/info/`.

JOB_DEP_LAST_SUB

Description Used only with job dependency scheduling.

If set to 1, whenever dependency conditions use a job name that belongs to multiple jobs, LSF evaluates only the most recently submitted job.

Otherwise, all the jobs with the specified name must satisfy the dependency condition.

Default Undefined

JOB_EXIT_RATE_DURATION

Description Defines how long LSF waits before checking the job exit rate for a host. Used in conjunction with EXIT_RATE in `lsb.hosts` for LSF host exception handling.

If the job exit rate is exceeded for the period specified by JOB_EXIT_RATE_DURATION, LSF invokes `LSF_SERVERDIR/eadmin` to trigger a host exception.

Example `JOB_EXIT_RATE_DURATION=5`

Default 10 minutes

JOB_POSITION_CONTROL_BY_ADMIN

Syntax **`JOB_POSITION_CONTROL_BY_ADMIN=Y | N`**

Description Allows LSF administrators to control whether users can use `btop` and `bbot` to move jobs to the top and bottom of queues. When JOB_POSITION_CONTROL_BY_ADMIN=Y, only the LSF administrator (including any queue administrators) can use `bbot` and `btop` to move jobs within a queue.

Default N

See also `bbot`(1), `btop`(10)

JOB_PRIORITY_OVER_TIME

Syntax **`JOB_PRIORITY_OVER_TIME=`***increment***`/`***interval*

Description JOB_PRIORITY_OVER_TIME enables automatic job priority escalation when MAX_USER_PRIORITY is also defined.

Valid Values *increment*

Specifies the value used to increase job priority every *interval* minutes. Valid values are positive integers.

interval

Specifies the frequency, in minutes, to *increment* job priority. Valid values are positive integers.

Default Undefined

Example JOB_PRIORITY_OVER_TIME=3/20

Specifies that every 20 minute *interval increment* to job priority of pending jobs by 3.

See also "MAX_USER_PRIORITY" on page 388.

JOB_SCHEDULING_INTERVAL

Syntax **`JOB_SCHEDULING_INTERVAL=`***seconds*

Description Time interval at which `mbatchd` sends jobs for scheduling to the scheduling daemon `mbschd` along with any collected load information.

If set to 0, there is no interval between job scheduling sessions.

Valid Value Number of seconds greater than or equal to zero (0).

Default 5 seconds

JOB_SPOOL_DIR

Syntax **`JOB_SPOOL_DIR=`***dir*

Description Specifies the directory for buffering batch standard output and standard error for a job.

When JOB_SPOOL_DIR is defined, the standard output and standard error for the job is buffered in the specified directory.

Files are copied from the submission host to a temporary file in the directory specified by the JOB_SPOOL_DIR on the execution host. LSF removes these files when the job completes.

If JOB_SPOOL_DIR is not accessible or does not exist, files are spooled to the default job output directory `$HOME/.lsbatch`.

For `bsub -is` and `bsub -Zs`, 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, and JOB_SPOOL_DIR is specified, `bsub -is` cannot write to the default directory `LSB_SHAREDIR/`*`cluster_name`*`/lsf_indir`, and `bsub -Zs` cannot write to the default directory `LSB_SHAREDIR/`*`cluster_name`*`/lsf_cmddir`, and the job will fail.

As LSF runs jobs, it creates temporary directories and files under JOB_SPOOL_DIR. By default, LSF removes these directories and files after the job is finished. See `bsub`(1) for information about job submission options that specify the disposition of these files.

On UNIX, specify an absolute path. For example:

```
JOB_SPOOL_DIR=/home/share/lsf_spool
```

On Windows, specify a UNC path or a path with a drive letter. For example:

```
JOB_SPOOL_DIR=\\HostA\share\spooldir
```

or

```
JOB_SPOOL_DIR=D:\share\spooldir
```

In a mixed UNIX/Windows cluster, specify one path for the UNIX platform and one for the Windows platform. Separate the two paths by a pipe character (|):

```
JOB_SPOOL_DIR=/usr/share/lsf_spool | \\HostA\share\spooldir
```

Valid value JOB_SPOOL_DIR can be any valid path up to a maximum length of 256 characters. This maximum path length includes the temporary directories and files that the LSF system creates as jobs run. The path you specify for JOB_SPOOL_DIR should be as short as possible to avoid exceeding this limit.

Default Undefined

Batch job output (standard output and standard error) is sent to the `.lsbatch` directory on the execution host:

- On UNIX: `$HOME/.lsbatch`
- On Windows: `%windir%\lsbtmp`*`user_id`*`\.lsbatch`

  If %HOME% is specified in the user environment, uses that directory instead of %windir% for spooled output.

JOB_TERMINATE_INTERVAL

Syntax **`JOB_TERMINATE_INTERVAL=`***seconds*

Description UNIX only.

Specifies the time interval in seconds between sending SIGINT, SIGTERM, and SIGKILL when terminating a job. When a job is terminated, the job is sent SIGINT, SIGTERM, and SIGKILL in sequence with a sleep time of JOB_TERMINATE_INTERVAL between sending the signals. This allows the job to clean up if necessary.

Default 10

MAX_ACCT_ARCHIVE_FILE

Syntax **MAX_ACCT_ARCHIVE_FILE=***integer*

Description Enables automatic deletion of archived LSF accounting log files and specifies the archive limit.

Compatibility ACCT_ARCHIVE_SIZE or ACCT_ARCHIVE_AGE should also be defined.

Example `MAX_ACCT_ARCHIVE_FILE=10`

LSF maintains the current `lsb.acct` and up to 10 archives. Every time the old `lsb.acct.9` becomes `lsb.acct.10`, the old `lsb.acct.10` gets deleted.

See also

- ACCT_ARCHIVE_AGE also enables automatic archiving.
- ACCT_ARCHIVE_SIZE also enables automatic archiving.
- ACCT_ARCHIVE_TIME also enables automatic archiving.
- MAX_ACCT_ARCHIVE_FILE enables automatic deletion of the archives.

Default Undefined (no deletion of `lsb.acct.`*n* files).

MAX_CONCURRENT_JOB_QUERY

Syntax **MAX_CONCURRENT_JOB_QUERY=***integer*

Description Defines how many concurrent job queries `mbatchd` can handle.

- If `mbatchd` is using multithreading, a dedicated query port is defined by the parameter `LSB_QUERY_PORT` in `lsf.conf`. When `mbatchd` has a dedicated query port, the value of `MAX_CONCURRENT_JOB_QUERY` sets the maximum number of job queries for each child `mbatchd` that is forked by `mbatchd`. This means that the total number of job queries can be more than the number specified by `MAX_CONCURRENT_JOB_QUERY`
- If `mbatchd` is not using multithreading, the value of `MAX_CONCURRENT_JOB_QUERY` sets the maximum total number of job queries.

Valid values 1-100

Default Unlimited

See Also `LSB_QUERY_PORT`

MAX_INFO_DIRS

Syntax **MAX_INFO_DIRS=***num_subdirs*

Description The number of subdirectories under the `LSB_SHAREDIR/`*`cluster_name`*`/logdir/info` directory.

When `MAX_INFO_DIRS` is enabled, `mbatchd` creates the specified number of subdirectories in the `info` directory. These subdirectories are given an integer as its name, starting with `0` for the first subdirectory. `mbatchd` writes the job files of all new submitted jobs into these subdirectories using the following formula to choose the subdirectory in which to store the job file:

```
subdirectory = jobID % MAX_INFO_DIRS
```

This formula ensures an even distribution of job files across the subdirectories.

IMPORTANT **If you are using local duplicate event logging, you must run badmin mbdrestart after changing MAX_INFO_DIRS for the changes to take effect.**

Valid values 1-1024

Default Not defined (no subdirectories under the `info` directory; `mbatchd` writes all jobfiles to the `info` directory)

Example MAX_INFO_DIRS=10

`mbatchd` creates ten subdirectories from `LSB_SHAREDIR/`*`cluster_name`*`/logdir/info/0` to `LSB_SHAREDIR/`*`cluster_name`*`/logdir/info/9`.

MAX_JOB_ARRAY_SIZE

Syntax **`MAX_JOB_ARRAY_SIZE=`***integer*

Description Specifies the maximum number of jobs in a job array that can be created by a user for a single job submission. The maximum number of jobs in a job array cannot exceed this value.

A large job array allows a user to submit a large number of jobs to the system with a single job submission.

Valid values Specify a positive integer between 1 and 2147483646

Default 1000

MAX_JOB_ATTA_SIZE

Syntax **`MAX_JOB_ATTA_SIZE=`***integer* | **`0`**

Specify any number less than 20000.

Description Maximum attached data size, in KB, that can be transferred to a job.

Maximum size for data attached to a job with the `bpost`(1) command. Useful if you use `bpost`(1) and `bread`(1) to transfer large data files between jobs and you want to limit the usage in the current working directory.

0 indicates that jobs cannot accept attached data files.

Default Undefined. LSF does not set a maximum size of job attachments.

MAX_JOBID

Syntax **`MAX_JOBID=`***integer*

Description The job ID limit. The job ID limit is the highest job ID that LSF will ever assign, and also the maximum number of jobs in the system.

By default, LSF assigns job IDs up to 6 digits. This means that no more than 999999 jobs can be in the system at once.

Specify any integer from 999999 to 9999999 (for practical purposes, any seven-digit integer).

You cannot lower the job ID limit, but you can raise it to seven digits. This means you can have more jobs in the system, and the job ID numbers will roll over less often.

LSF assigns job IDs in sequence. When the job ID limit is reached, the count rolls over, so the next job submitted gets job ID "1". If the original job 1 remains in the system, LSF skips that number and assigns job ID "2", or the next available job ID. If you have so many jobs in the system that the low job IDs are still in use when the maximum job ID is assigned, jobs with sequential numbers could have totally different submission times.

By raising the job ID limit, you allow more time for old jobs to leave the system, and make it more likely that numbers can be assigned in sequence without conflicting with existing jobs.

Example MAX_JOBID=1234567

Default 999999

MAX_JOBINFO_QUERY_PERIOD

Syntax **MAX_JOBINFO_QUERY_PERIOD=***integer*

Description Maximum time for job information query commands (for example, with `bjobs`) to wait.

When the time arrives, the query command processes exit, and all associated threads are terminated.

If the parameter is not defined, query command processes will wait for all threads to finish.

Specify a multiple of MBD_REFRESH_TIME.

Valid values Any positive integer greater than or equal to one (1)

Default Undefined

See also See "lsf.conf" under "LSB_BLOCK_JOBINFO_TIMEOUT" on page 509.

MAX_JOB_MSG_NUM

Syntax **MAX_JOB_MSG_NUM=***integer* | **0**

Description Maximum number of message slots for each job. Maximum number of messages that can be posted to a job with the `bpost`(1) command.

0 indicates that jobs cannot accept external messages.

Default 128

MAX_JOB_NUM

Syntax **MAX_JOB_NUM=***integer*

Description The maximum number of finished jobs whose events are to be stored in the `lsb.events` log file.

Once the limit is reached, `mbatchd` starts a new event log file. The old event log file is saved as `lsb.events.`*n*, with subsequent sequence number suffixes incremented by 1 each time a new log file is started. Event logging continues in the new `lsb.events` file.

Default 1000

MAX_PEND_JOBS

Syntax **MAX_PEND_JOBS=***integer*

Description The maximum number of pending jobs in the system.

This is the hard system-wide pending job threshold. No user or user group can exceed this limit unless the job is forwarded from a remote cluster.

If the user or user group submitting the job has reached the pending job threshold as specified by `MAX_PEND_JOBS`, LSF will reject any further job submission requests sent by that user or user group. The system will continue to send the job submission requests with the interval specified by `SUB_TRY_INTERVAL` in `lsb.params` until it has made a number of attempts equal to the `LSB_NTRIES` environment variable. If `LSB_NTRIES` is undefined and LSF rejects the job submission request, the system will continue to send the job submission requests indefinitely as the default behavior.

Default INFINIT_INT

INFINIT_INT is defined in `lsf.h`

See also SUB_TRY_INTERVAL

MAX_PREEXEC_RETRY

Syntax **MAX_PREEXEC_RETRY=***integer*

Description MultiCluster job forwarding model only. The maximum number of times to attempt the pre-execution command of a job from a remote cluster.

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster.

MAX_SBD_CONNS

Syntax **MAX_SBD_CONNS=***integer*

Description The maximum number of file descriptors `mbatchd` can have open and connected concurrently to `sbatchd`

Controls the maximum number of connections that LSF can maintain to `sbatchd`s in the system. Many sites require more than 32 connections.

Do not exceed the file descriptor limit of the root process (the usual limit is 1024). Setting it equal or larger than this limit can cause `mbatchd` to constantly die because `mbatchd` allocates all file descriptors to `sbatchd` connection. This could cause `mbatchd` to run out of descriptors, which results in an `mbatchd` fatal error, such as failure to open `lsb.events`.

Example Reasonable settings are:

- `MAX_SBD_CONNS=512`
- `MAX_SBD_CONNS=768`

Default 32

MAX_SBD_FAIL

Syntax **MAX_SBD_FAIL=***integer*

Description The maximum number of retries for reaching a non-responding slave batch daemon, `sbatchd`.

The interval between retries is defined by MBD_SLEEP_TIME. If `mbatchd` fails to reach a host and has retried MAX_SBD_FAIL times, the host is considered unreachable. When a host becomes unreachable, `mbatchd` assumes that all jobs running on that host have exited and that all rerunnable jobs (jobs submitted with the `bsub -r` option) are scheduled to be rerun on another host.

Default 3

MAX_SCHED_STAY (OBSOLETE)

Syntax **`MAX_SCHED_STAY=`***integer*

Description This parameter is obsolete.

Default 3

MAX_USER_PRIORITY

Syntax **`MAX_USER_PRIORITY=`***integer*

Description Enables user-assigned job priority and specifies the maximum job priority a user can assign to a job.

LSF administrators can assign a job priority higher than the specified value.

Compatibility User-assigned job priority changes the behavior of `btop` and `bbot`.

Example `MAX_USER_PRIORITY=100`

Specifies that 100 is the maximum job priority that can be specified by a user.

Default Undefined

See also

- `bsub`, `bmod`, `btop`, `bbot`
- "JOB_PRIORITY_OVER_TIME" on page 382.

MBD_REFRESH_TIME

Syntax **`MBD_REFRESH_TIME=`***seconds* [*min_refresh_time*]

where

- *min_refresh_time* defines the minimum time (in seconds) that the child mbatchd will stay to handle queries. The valid range is 0 - 300 (MAX_MBD_REFRESH_TIME) with 10 as default.

Description Time interval, in seconds, at which `mbatchd` will fork a new child `mbatchd` to service query requests to keep information sent back to clients updated. A child `mbatchd` processes query requests creating threads.

MBD_REFRESH_TIME applies only to UNIX platforms that support thread programming.

To enable MBD_REFRESH_TIME you must specify LSB_QUERY_PORT in `lsf.conf`. The child `mbatchd` listens to the port number specified by LSB_QUERY_PORT and creates threads to service requests until the job changes status, a new job is submitted, or MBD_REFRESH_TIME has expired.

- If MBD_REFRESH_TIME is < 10 seconds, the child `mbatchd` exits at MBD_REFRESH_TIME if the job changes status or a new job is submitted before MBD_REFRESH_TIME expires
- If MBD_REFRESH_TIME > 10 seconds, the child `mbatchd` exits at 10 seconds if the job changes status or a new job is submitted before the 10 seconds
- If MBD_REFRESH_TIME > 10 seconds and no job changes status or no new job is submitted, the child `mbatchd` exits at MBD_REFRESH_TIME

The value of this parameter must be between 5 and 300. Any values specified out of this range are ignored, and the system default value is applied.

The `bjobs` command may not display up-to-date information if two consecutive query commands are issued before a child `mbatchd` expires because child `mbatchd` job information is not updated. If you use the `bjobs` command and do not get up-to-date information, you may need to decrease the value of this parameter. Note, however, that the lower the value of this parameter, the more you negatively affect performance.

The number of concurrent requests is limited by the number of concurrent threads that a process can have. This number varies by platform:

- Sun Solaris, 2500 threads per process
- AIX, 512 threads per process
- Digital, 256 threads per process
- HP-UX, 64 threads per process

Default 5 seconds if not defined or if defined value is less than 5; 300 seconds if defined value is more than 300

MBD_SLEEP_TIME

Syntax **MBD_SLEEP_TIME=***seconds*

Description Used in conjunction with the parameters SLOT_RESERVE, MAX_SBD_FAIL, and JOB_ACCEPT_INTERVAL.

Amount of time in seconds used for calculating parameter values.

Default 60

MC_RECLAIM_DELAY

Syntax **MC_RECLAIM_DELAY=***minutes*

Description MultiCluster resource leasing model only. The reclaim interval (how often to reconfigure shared leases) in minutes.

Shared leases are defined by `Type=shared` in the `lsb.resources` HostExport section.

Default 10

MC_PENDING_REASON_PKG_SIZE

Syntax **MC_PENDING_REASON_PKG_SIZE=***kilobytes* | **0**

Description MultiCluster job forwarding model only. Pending reason update package size, in KB. Defines the maximum amount of pending reason data this cluster will send to submission clusters in one cycle.

Specify the keyword `0` (zero) to disable the limit and allow any amount of data in one package.

Default 512

MC_PENDING_REASON_UPDATE_INTERVAL

Syntax **MC_PENDING_REASON_UPDATE_INTERVAL=***seconds* | **0**

Description MultiCluster job forwarding model only. Pending reason update interval, in seconds. Defines how often this cluster will update submission clusters about the status of pending MultiCluster jobs.

Specify the keyword `0` (zero) to disable pending reason updating between clusters.

Default 300

MC_RUSAGE_UPDATE_INTERVAL

Syntax **MC_RUSAGE_UPDATE_INTERVAL=***seconds*

Description MultiCluster only. Enables resource use updating for MultiCluster jobs running on hosts in the cluster and specifies how often to send updated information to the submission or consumer cluster.

Default 300

MIN_SWITCH_PERIOD

Syntax **MIN_SWITCH_PERIOD=***seconds*

Description The minimum period in seconds between event log switches.

Works together with `MAX_JOB_NUM to` control how frequently `mbatchd` switches the file. `mbatchd` checks if `MAX_JOB_NUM` has been reached every `MIN_SWITCH_PERIOD` seconds. If `mbatchd` finds that `MAX_JOB_NUM` has been reached, it switches the events file.

Default 0

No minimum period. Log switch frequency is not restricted.

See Also `MAX_PEND_JOBS`

NO_PREEMPT_RUN_TIME

Syntax **NO_PREEMPT_RUN_TIME=***run_time*

Description If set, jobs have been running for the specified number of minutes or longer will not be preempted. Run time is wall-clock time, not normalized run time.

You must define a run limit for the job, either at job level by `bsub -W` option or in the queue by configuring RUNLIMIT in `lsb.queues`.

NO_PREEMPT_FINISH_TIME

Syntax **NO_PREEMPT_FINISH_TIME=***finish_time*

Description If set, jobs that will finish within the specified number of minutes will not be preempted. Run time is wall-clock time, not normalized run time.

You must define a run limit for the job, either at job level by `bsub -W` option or in the queue by configuring RUNLIMIT in `lsb.queues`.

NQS_QUEUES_FLAGS

Syntax **NQS_QUEUES_FLAGS=***integer*

Description For Cray NQS compatibility only. Used by LSF to get the NQS queue information.

If the NQS version on a Cray is NQS 1.1, 80.42 or NQS 71.3, this parameter does not need to be defined.

For other versions of NQS on Cray, define both NQS_QUEUES_FLAGS and NQS_REQUESTS_FLAGS.

To determine the value of this parameter, run the NQS `qstat` command. The value of `Npk_int[1]` in the output is the value you need for this parameter. Refer to the NQS chapter in *Administering Platform LSF* for more details.

Default Undefined

NQS_REQUESTS_FLAGS

Syntax **NQS_REQUESTS_FLAGS=***integer*

Description For Cray NQS compatibility only.

If the NQS version on a Cray is NQS 80.42 or NQS 71.3, this parameter does not need to be defined.

If the version is NQS 1.1 on a Cray, set this parameter to 251918848. This is the is the `qstat` flag which LSF uses to retrieve requests on Cray in long format.

For other versions of NQS on a Cray, run the NQS `qstat` command. The value of `Npk_int[1]` in the output is the value you need for this parameter. Refer to the NQS chapter in *Administering Platform LSF* for more details.

Default Undefined

PARALLEL_SCHED_BY_SLOT

Syntax **PARALLEL_SCHED_BY_SLOT=y | Y**

Description If defined, LSF schedules jobs based on the number of slots assigned to the hosts instead of the number of CPUs. These slots can be defined by host in `lsb.hosts` or by slot limit in `lsb.resources`.

All slot-related messages still show the word “processors”, but actually refer to “slots” instead. Similarly, all scheduling activities also use slots instead of processors.

Default Undefined

See also
- See JL/U and MXJ under “lsb.hosts” on page 353

- See SLOTS and SLOTS_PER_PROCESSOR under “lsb.resources” on page 433

PEND_REASON_UPDATE_INTERVAL

Syntax **PEND_REASON_UPDATE_INTERVAL=***seconds*

Description Time interval that defines how often pending reasons are calculated by the scheduling daemon `mbschd`.

Default 30 seconds

PEND_REASON_MAX_JOBS

Syntax **PEND_REASON_MAX_JOBS=***integer*

Description Number of jobs for each user per queue for which pending reasons are calculated by the scheduling daemon `mbschd`. Pending reasons are calculated at a time period set by PEND_REASON_UPDATE_INTERVAL.

Default 20 jobs

PG_SUSP_IT

Syntax **PG_SUSP_IT=***seconds*

Description The time interval that a host should be interactively idle (it > 0) before jobs suspended because of a threshold on the `pg` load index can be resumed.

This parameter is used to prevent the case in which a batch job is suspended and resumed too often as it raises the paging rate while running and lowers it while suspended. If you are not concerned with the interference with interactive jobs caused by paging, the value of this parameter may be set to 0.

Default 180 (seconds)

PREEMPTABLE_RESOURCES

Syntax **PREEMPTABLE_RESOURCES=***resource_name* ...

Description LicenseMaximizer only. Enables license preemption when preemptive scheduling is enabled (has no effect if PREEMPTIVE is not also specified) and specifies the licenses that will be preemption resources. Specify shared numeric resources, static or decreasing, that LSF is configured to release (RELEASE=Y in `lsf.shared`, which is the default).

You must also configure LSF preemption actions to make the preempted application releases its licenses. To kill preempted jobs instead of suspending them, set TERMINATE_WHEN=PREEMPT in `lsb.queues`, or set JOB_CONTROLS in `lsb.queues` and specify `brequeue` as the SUSPEND action.

Default Undefined (if preemptive scheduling is configured, LSF preempts on job slots only)

PREEMPT_FOR

Syntax **PREEMPT_FOR=**[**HOST_JLU** | **USER_JLP** | **GROUP_MAX** | **GROUP_JLP** | **MINI_JOB** |**LEAST_RUN_TIME**]...

Description If preemptive scheduling is enabled, this parameter can change the behavior of job slot limits and can also enable the optimized preemption mechanism for parallel jobs.

Specify a space-separated list of the following keywords:

- GROUP_MAX—LSF does not count suspended jobs against the total job slot limit for user groups, specified at the user level (MAX_JOBS in `lsb.users`); if preemptive scheduling is enabled, suspended jobs never count against the limit for individual users
- HOST_JLU—LSF does not count suspended jobs against the total number of jobs for users and user groups, specified at the host level (JL/U in `lsb.hosts`)
- USER_JLP—LSF does not count suspended jobs against the user-processor job slot limit for individual users, specified at the user level (JL/P in `lsb.users`)
- GROUP_JLP—LSF does not count suspended jobs against the per-processor job slot limit for user groups, specified at the user level (JL/P in `lsb.users`)
- MINI_JOB—LSF uses the optimized preemption mechanism for preemption between parallel jobs
- LEAST_RUN_TIME—LSF preempts job with least run time. Run time is wall-clock time, not normalized run time.

Job slot limits specified at the queue level always count suspended jobs.

Default Undefined. If preemptive scheduling is configured, the default preemption mechanism is used to preempt parallel jobs, and suspended jobs are ignored for the following limits only:

- Total job slot limit for hosts, specified at the host level (MXJ in `lsb.hosts`)
- Total job slot limit for individual users, specified at the user level (MAX_JOBS in `lsb.users`); by default, suspended jobs still count against the limit for user groups

PREEMPTION_WAIT_TIME

Syntax **PREEMPTION_WAIT_TIME=***seconds*

Description LicenseMaximizer only. You must also specify PREEMPTABLE_RESOURCES in `lsb.params`).

The amount of time LSF waits, after preempting jobs, for preemption resources to become available. Specify at least 300 seconds.

If LSF does not get the resources after this time, LSF might preempt more jobs.

Default 300 (5 minutes)

RESOURCE_RESERVE_PER_SLOT

Syntax **RESOURCE_RESERVE_PER_SLOT=y | Y**

Description If Y, `mbatchd` reserves resources based on job slots instead of per-host.

By default, `mbatchd` only reserves static resources for parallel jobs on a per-host basis. For example, by default, the command:

```
% bsub -n 4 -R "rusage[mem=500]" -q reservation my_job
```

requires the job to reserve 500 MB on each host where the job runs.

Some parallel jobs need to reserve resources based on job slots, rather than by host. In this example, if per-slot reservation is enabled by RESOURCE_RESERVE_PER_SLOT, the job `my_job` must reserve 500 MB of memory for each job slot (4*500=2 GB) on the host in order to run.

If RESOURCE_RESERVE_PER_SLOT is set, the following command reserves the resource `static_resource` on all 4 job slots instead of only 1 on the host where the job runs:

```
bsub -n 4 -R "static_resource > 0 rusage[static_resource=1]" 
myjob
```

Default Undefined (reserve resources per-host)

RUN_JOB_FACTOR

Syntax **RUN_JOB_FACTOR=***number*

Description Used only with fairshare scheduling. Job slots weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of job slots reserved and in use by a user.

Default 3.0

RUN_TIME_FACTOR

Syntax **RUN_TIME_FACTOR=***number*

Description Used only with fairshare scheduling. Run time weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the total run time of a user's running jobs.

Default 0.7

SBD_SLEEP_TIME

Syntax **SBD_SLEEP_TIME=***seconds*

Description The interval at which LSF checks the load conditions of each host, to decide whether jobs on the host must be suspended or resumed.

The job-level resource usage information is updated at a maximum frequency of every SBD_SLEEP_TIME seconds.

The update is done only if the value for the CPU time, resident memory usage, or virtual memory usage has changed by more than 10 percent from the previous update or if a new process or process group has been created.

Default 30

SUB_TRY_INTERVAL

Syntax **SUB_TRY_INTERVAL=***integer*

Description The number of seconds for the requesting client to wait before resubmitting a job. This is sent by `mbatchd` to the client.

Default 60

See also "MAX_PEND_JOBS" on page 387

SYSTEM_MAPPING_ACCOUNT

Syntax **SYSTEM_MAPPING_ACCOUNT=***user_account*

Description LSF Windows Workgroup installations only. User account to which all Windows workgroup user accounts are mapped.

Default Undefined

Automatic Time-based Configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.params` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the `badmin reconfig` command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

```
# if 18:30-19:30 is your short job express period, but
# you want all jobs going to the short queue by default
# and be subject to the thresholds of that queue

# for all other hours, normal is the default queue

#if time(18:30-19:30)
DEFAULT_QUEUE=short
#else
DEFAULT_QUEUE=normal
#endif
```

SEE ALSO

lsf.conf(5), lsb.params(5), lsb.hosts(5), lsb.users(5), bsub(1)

lsb.queues

The `lsb.queues` file defines batch queues. Numerous controls are available at the queue level to allow cluster administrators to customize site policies.

This file is optional; if no queues are configured, LSF creates a queue named `default`, with all parameters set to default values.

This file is installed by default in `LSB_CONFDIR/`*`cluster_name`*`/configdir`.

Changing lsb.queues configuration

After making any changes to `lsb.queues`, run `badmin reconfig` to reconfigure `mbatchd`.

Contents

- "lsb.queues Structure" on page 400
- "Automatic Time-based Configuration" on page 430

lsb.queues Structure

Each queue definition begins with the line Begin Queue and ends with the line End Queue. The queue name must be specified; all other parameters are optional.

ADMINISTRATORS

Syntax **ADMINISTRATORS=***user_name* | *user_group* ...

Description List of queue administrators.

Queue administrators can perform operations on any user's job in the queue, as well as on the queue itself.

Default Undefined (you must be a cluster administrator to operate on this queue)

BACKFILL

Syntax **BACKFILL=Y** | **N**

Description If Y, enables backfill scheduling for the queue.

A possible conflict exists if BACKFILL and PREEMPTION are specified together. A backfill queue cannot be preemptable. Therefore, if BACKFILL is enabled, do not also specify PREEMPTION=PREEMPTABLE.

BACKFILL is required for interruptible backfill queues (INTERRUPTIBLE_BACKFILL=*seconds*).

Default Undefined (no backfilling)

CHKPNT

Syntax **CHKPNT=***chkpnt_dir* [*chkpnt_period*]

Description Enables automatic checkpointing for the queue.

The checkpoint directory is the directory where the checkpoint files are created. Specify an absolute path or a path relative to CWD, do not use environment variables.

Specify the optional checkpoint period in minutes.

Job-level checkpoint parameters override queue-level checkpoint parameters.

Only running members of a chunk job can be checkpointed.

To make a MultiCluster job checkpointable, both submission and execution queues must enable checkpointing, and the execution queue setting determines the checkpoint directory. Checkpointing is not supported if a job runs on a leased host.

Default Undefined

CHUNK_JOB_SIZE

Syntax **CHUNK_JOB_SIZE=***integer*

Description Chunk jobs only. Enables job chunking and specifies the maximum number of jobs allowed to be dispatched together in a chunk. Specify a positive integer greater than 1.

The ideal candidates for job chunking are jobs that have the same host and resource requirements and typically take 1 to 2 minutes to run.

Job chunking can have the following advantages:

- Reduces communication between `sbatchd` and `mbatchd` and reduces scheduling overhead in `mbschd`.
- Increases job throughput in `mbatchd` and CPU utilization on the execution hosts.

However, throughput can deteriorate if the chunk job size is too big. Performance may decrease on queues with CHUNK_JOB_SIZE greater than 30. You should evaluate the chunk job size on your own systems for best performance.

With MultiCluster job forwarding model, this parameter does not affect MultiCluster jobs that are forwarded to a remote cluster.

Compatibility

This parameter is ignored in the following kinds of queues:

- Interactive (INTERACTIVE=ONLY parameter)
- CPU limit greater than 30 minutes (CPULIMIT parameter)
- Run limit greater than 30 minutes (RUNLIMIT parameter)

If CHUNK_JOB_DURATION is set in `lsb.params`, chunk jobs are accepted regardless of the value of CPULIMIT or RUNLIMIT.

Example

The following configures a queue named `chunk`, which dispatches up to 4 jobs in a chunk:

```
Begin Queue
QUEUE_NAME     = chunk
PRIORITY       = 50
CHUNK_JOB_SIZE = 4
End Queue
```

Default

Undefined

CORELIMIT

Syntax

CORELIMIT=*integer*

Description

The per-process (hard) core file size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

Default

Unlimited

CPULIMIT

Syntax

CPULIMIT=[*default_limit*] *maximum_limit*

where *default_limit* and *maximum_limit* are:

[*hour***:**]*minute*[**/***host_name* | **/***host_model*]

Description

Maximum normalized CPU time and optionally, the default normalized CPU time allowed for all processes of a job running in this queue. The name of a host or host model specifies the CPU time normalization host to use.

Limits the total CPU time the job can use. This parameter 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 sent to all processes belonging 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 to the job to kill it.

If a job dynamically spawns processes, the CPU time used by these processes is accumulated over the life of the job.

Processes that exist for fewer than 30 seconds may be ignored.

By default, if a default CPU limit is specified, jobs submitted to the queue without a job-level CPU limit are killed when the default CPU limit is reached.

If you specify only one limit, it is the maximum, or hard, CPU limit. If you specify two limits, the first one is the default, or soft, CPU limit, and the second one is the maximum CPU limit. The number of minutes may be greater than 59. Therefore, three and a half hours can be specified either as 3:30 or 210.

If no host or host model is given with the CPU time, 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 host with the largest CPU factor (the fastest host in the cluster).

On Windows, a job which runs under a CPU time limit may exceed that limit by up to SBD_SLEEP_TIME. This is because `sbatchd` periodically checks if the limit has been exceeded.

On UNIX systems, the CPU limit can be enforced by the operating system at the process level.

You can define whether the CPU limit is a per-process limit enforced by the OS or a per-job limit enforced by LSF with LSB_JOB_CPULIMIT in `lsf.conf`.

Jobs submitted to a chunk job queue are not chunked if CPULIMIT is greater than 30 minutes.

Default Unlimited

DATALIMIT

Syntax **DATALIMIT=**[*default_limit*] *maximum_limit*

Description The per-process data segment size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

By default, if a default data limit is specified, jobs submitted to the queue without a job-level data limit are killed when the default data limit is reached.

If you specify only one limit, it is the maximum, or hard, data limit. If you specify two limits, the first one is the default, or soft, data limit, and the second one is the maximum data limit

Default Unlimited

DEFAULT_EXTSCHED

Syntax **DEFAULT_EXTSCHED=***external_scheduler_options*

Description Specifies default external scheduling options for the queue.

`-extsched` options on the `bsub` command are merged with DEFAULT_EXTSCHED options, and `-extsched` options override any conflicting queue-level options set by DEFAULT_EXTSCHED.

Default Undefined

DEFAULT_HOST_SPEC

Syntax **DEFAULT_HOST_SPEC=***host_name* | *host_model*

Description The default CPU time normalization host for the queue.

The CPU factor of the specified host or host model will be used to normalize the CPU time limit of all jobs in the queue, unless the CPU time normalization host is specified at the job level.

Default Undefined

DESCRIPTION

Syntax **DESCRIPTION=***text*

Description Description of the job queue that will be displayed by `bqueues -l`.

This description should clearly describe the service features of this queue, to help users select the proper queue for each job.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 512 characters.

DISPATCH_ORDER

Syntax **DISPATCH_ORDER=QUEUE**

Description Defines an *ordered* cross-queue fairshare set. DISPATCH_ORDER indicates that jobs are dispatched according to the order of queue priorities first, then user fairshare priority.

By default, a user has the same priority across the master and slave queues. If the same user submits several jobs to these queues, user priority is calculated by taking into account all the jobs the user has submitted across the master-slave set.

If DISPATCH_ORDER=QUEUE is set in the master queue, jobs are dispatched according to queue priorities first, then user priority. Jobs from users with lower fairshare priorities who have pending jobs in higher priority queues are dispatched before jobs in lower priority queues. This avoids having users with higher fairshare priority getting jobs dispatched from low-priority queues.

Jobs in queues having the same priority are dispatched according to user priority.

Queues that are not part of the cross-queue fairshare can have any priority; the are not limited to fall outside of the priority range of cross-queue fairshare queues.

Default Undefined

DISPATCH_WINDOW

Syntax **DISPATCH_WINDOW=***time_window* ...

Description The time windows in which jobs from this queue are dispatched. Once dispatched, jobs are no longer affected by the dispatch window.

Default Undefined (always open)

EXCLUSIVE

Syntax **EXCLUSIVE=Y** | **N**

Description If Y, specifies an exclusive queue.

Jobs submitted to an exclusive queue with `bsub -x` will only be dispatched to a host that has no other LSF jobs running.

For hosts shared under the MultiCluster resource leasing model, jobs will not be dispatched to a host that has LSF jobs running, even if the jobs are from another cluster.

FAIRSHARE

Description Enables queue-level user-based fairshare and specifies share assignments. Only users with share assignments can submit jobs to the queue.

Syntax **FAIRSHARE=USER_SHARES[[***user***,** *number_shares***]** ...**]**

- Specify at least one user share assignment.
- Enclose the list in square brackets, as shown.
- Enclose each user share assignment in square brackets, as shown.
- *user*

  Specify users who are also configured to use queue. You can assign the shares to:

  - A single user (specify *user_name*)
  - Users in a group, individually (specify *group_name***@**) or collectively (specify *group_name*)
  - Users not included in any other share assignment, individually (specify the keyword `default`) or collectively (specify the keyword `others`)

  By default, when resources are assigned collectively to a group, the group members compete for the resources on a first-come, first-served (FCFS) basis. You can use hierarchical fairshare to further divide the shares among the group members.

  When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*

  Specify a positive integer representing the number of shares of the cluster resources assigned to the user.

The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

Compatibility Do not configure hosts in a cluster to use fairshare at both queue and host levels. However, you can configure user-based fairshare and queue-based fairshare together.

Default Undefined (no fairshare)

FAIRSHARE_QUEUES

Syntax **FAIRSHARE_QUEUES=***queue_name queue_name* ...

Description Defines cross-queue fairshare.

When this parameter is defined:

- The queue in which this parameter is defined becomes the "*master queue*".
- Queues listed with this parameter are "*slave queues*" and inherit the fairshare policy of the master queue.
- A user has the same priority across the master and slave queues.

  If the same user submits several jobs to these queues, user priority is calculated by taking into account all the jobs the user has submitted across the master-slave set.

Notes

- By default, the PRIORITY range defined for queues in cross-queue fairshare cannot be used with any other queues. For example, you have 4 queues: queue1, queue2, queue3, `queue4`. You configure cross-queue fairshare for queue1, queue2, queue3 and assign priorities of 30, 40, 50 respectively.
- By default, the priority of `queue4` (which is not part of the cross-queue fairshare) cannot fall between the priority range of the cross-queue fairshare queues (30-50). It can be any number up to 29 or higher than 50. It does not matter if `queue4` is a fairshare queue or FCFS queue.

  If DISPATCH_ORDER=QUEUE is set in the master queue, the priority of `queue4` (which is not part of the cross-queue fairshare) can be any number, including a priority falling between the priority range of the cross-queue fairshare queues (30-50).
- FAIRSHARE must be defined in the master queue. If it is also defined in the queues listed in FAIRSHARE_QUEUES, it will be ignored.
- Cross-queue fairshare can be defined more than once within `lsb.queues`. You can define several sets of master-slave queues. However, a queue cannot belong to more than one master-slave set. For example, you can define:
  - In queue `normal`: `FAIRSHARE_QUEUES=short license`
  - In queue `priority`: `FAIRSHARE_QUEUES=night owners`

  You cannot, however, define `night`, `owners`, or `priority` as slaves in the queue `normal`; or `normal`, `short` and `license` as slaves in the `priority` queue; or `short`, `license`, `night`, `owners` as master queues of their own.
- Cross-queue fairshare cannot be used with host partition fairshare. It is part of queue-level fairshare.

Default Undefined

FILELIMIT

Syntax **`FILELIMIT=`***integer*

Description The per-process (hard) file size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

Default Unlimited

HJOB_LIMIT

Syntax **`HJOB_LIMIT`**=*integer*

Description Per-host job slot limit.

Maximum number of job slots that this queue can use on any host. This limit is configured per host, regardless of the number of processors it may have.

This may be useful if the queue dispatches jobs that require a node-locked license. If there is only one node-locked license per host then the system should not dispatch more than one job to the host even if it is a multiprocessor host.

Example The following will run a maximum of one job on each of `hostA`, `hostB`, and `hostC`:

```
Begin Queue
...
HJOB_LIMIT = 1
HOSTS=hostA hostB hostC
...
End Queue
```

Default Unlimited

HOSTS

Syntax **`HOSTS=`***host_list* | **`none`**

- *host_list* is a space-separated list of the following items:
  - *host_name*[**@***cluster_name*][**+***pref_level*]
  - *host_partition*[**+***pref_level*]
  - *host_group*[**+***pref_level*]
  - [**~**]*host_name*
  - [**~**]*host_group*
  - **`all@`***cluster_name*

  The list can include the following items only once:
  - **`others`**[**+***pref_level*]
  - **`all`**
  - **`allremote`**
- **`none`** keyword is only used with the MultiCluster job forwarding model, to specify a remote-only queue.

Description A space-separated list of hosts on which jobs from this queue can be run.

If host groups and host partitions are included in the list, the job can run on any host in the group or partition. All the members of the host list should either belong to a single host partition or not belong to any host partition. Otherwise, job scheduling may be affected.

Some items can be followed by a plus sign (+) and a positive number to indicate the preference for dispatching a job to that host. A higher number indicates a higher preference. If a host preference is not given, it is assumed to be 0. If there are multiple candidate hosts, LSF dispatches the job to the host with the highest preference; hosts at the same level of preference are ordered by load.

If host groups and host partitions are assigned a preference, each host in the group or partition has the same preference.

Use the keyword `others` to include all hosts not explicitly listed.

Use the keyword `all` to include all hosts not explicitly excluded.

Use the keyword `all@`*cluster_name hostgroup_name* or `allremote` *hostgroup_name* to include lease in hosts.

Use the not operator (~) to exclude hosts from the `all` specification in the queue. This is useful if you have a large cluster but only want to exclude a few hosts from the queue definition.

The not operator can only be used with the `all` keyword. It is *not* valid with the keywords `others` and `none`.

The not operator (~) can be used to exclude host groups.

With MultiCluster resource leasing model, use the format *host_name@cluster_name* to specify a borrowed host. LSF does not validate the names of remote hosts. The keyword `others` indicates all local hosts not explicitly listed. The keyword `all` indicates all local hosts not explicitly excluded. Use the keyword `allremote` to specify all hosts borrowed from all remote clusters. Use `all@`*cluster_name* to specify the group of all hosts borrowed from one remote cluster. You cannot specify a host group or partition that includes remote resources, unless it uses the keyword `allremote` to include all remote hosts.

With MultiCluster resource leasing model, the not operator (~) can be used to exclude local hosts or host groups. You cannot use the not operator (~) with remote hosts.

Hosts that participate in queue-based fairshare cannot be in a host partition.

Compatibility

Host preferences specified by `bsub -m` override the queue specification.

Example 1

`HOSTS=hostA+1 hostB hostC+1 hostD+3`

This example defines three levels of preferences: run jobs on `hostD` as much as possible, otherwise run on either `hostA` or `hostC` if possible, otherwise run on `hostB`. Jobs should not run on `hostB` unless all other hosts are too busy to accept more jobs.

Example 2

`HOSTS=hostD+1 others`

Run jobs on `hostD` as much as possible, otherwise run jobs on the least-loaded host available.

With MultiCluster resource leasing model, this queue does not use borrowed hosts.

Example 3

`HOSTS=all ~hostA`

Run jobs on all hosts in the cluster, except for `hostA`.

With MultiCluster resource leasing model, this queue does not use borrowed hosts.

Example 4 `HOSTS=Group1 ~hostA hostB hostC`

Run jobs on `hostB`, `hostC`, and all hosts in `Group1` except for `hostA`.

With MultiCluster resource leasing model, this queue will use borrowed hosts if Group1 uses the keyword `allremote`.

Default `all` (the queue can use all hosts in the cluster, and every host has equal preference)

With MultiCluster resource leasing model, this queue can use all local hosts, but no borrowed hosts.

IGNORE_DEADLINE

Syntax **`IGNORE_DEADLINE=Y`**

Description If Y, disables deadline constraint scheduling (starts all jobs regardless of deadline constraints).

IMPT_JOBBKLG

Syntax **`IMPT_JOBBKLG=`***integer* | **`infinit`**

Description MultiCluster job forwarding model only. Specifies the MultiCluster pending job limit for a receive-jobs queue. This represents the maximum number of MultiCluster jobs that can be pending in the queue; once the limit has been reached, the queue stops accepting jobs from remote clusters.

Use the keyword `infinit` to make the queue accept an infinite number of pending MultiCluster jobs.

Default 50

INTERACTIVE

Syntax **`INTERACTIVE=NO`** | **`ONLY`**

Description Causes the queue to reject interactive batch jobs (NO) or accept nothing but interactive batch jobs (ONLY).

Interactive batch jobs are submitted via `bsub -I`.

Default Undefined (the queue accepts both interactive and non-interactive jobs)

INTERRRUPTIBLE_BACKFILL

Syntax **`INTERRRUPTIBLE_BACKFILL=`***seconds*

Description Configures interruptible backfill scheduling policy, which allows reserved job slots to be used by low priority small jobs that will be terminated when the higher priority large jobs are about to start.

There can only be one interruptible backfill queue.It should be the lowest priority queue in the cluster.

Specify the minimum number of seconds for the job to be considered for backfilling.This minimal time slice depends on the specific job properties; it must be longer than at least one useful iteration of the job. Multiple queues may be created if a site has jobs of distinctively different classes.

An interruptible backfill job:

- Starts as a regular job and is killed when it exceeds the queue runtime limit

OR

- Is started for backfill whenever there is a backfill time slice longer than the specified minimal time, and killed before the slot-reservation job is about to start

The queue RUNLIMIT corresponds to a maximum time slice for backfill, and should be configured so that the wait period for the new jobs submitted to the queue is acceptable to users. 10 minutes of runtime is a common value.

You should configure REQUEUE_EXIT_VALUES for interruptible backfill queues.

BACKFILL and RUNLIMIT must be configured in the queue. The queue is disabled if BACKFILL and RUNLIMIT are not configured.

Assumptions and limitations:

- The interruptible backfill job will hold the slot-reserving job start until its calculated start time, in the same way as a regular backfill job. The interruptible backfill job will not be preempted in any way other than being killed when its time come.
- While the queue is checked for the consistency of interruptible backfill, backfill and runtime specifications, the requeue exit value clause is not verified, nor executed automatically. Configure requeue exit values according to your site policies.
- The interruptible backfill job must be able to do at least one unit of useful calculations and save its data within the minimal time slice, and be able to continue its calculations after it has been restarted
- Interruptible backfill paradigm does not explicitly prohibit running parallel jobs, distributed across multiple nodes, however, the chance of success of such job is close to zero.

Default

Undefined (no interruptible backfilling)

JOB_ACCEPT_INTERVAL

Syntax

JOB_ACCEPT_INTERVAL=*integer*

Description

The number you specify is multiplied by the value of `lsb.params` MBD_SLEEP_TIME (60 seconds by default). The result of the calculation is the number of seconds to wait after dispatching a job to a host, before dispatching a second job to the same host.

If 0 (zero), a host may accept more than one job in each dispatch turn. By default, there is no limit to the total number of jobs that can run on a host, so if this parameter is set to 0, a very large number of jobs might be dispatched to a host all at once. This can overload your system to the point that it will be unable to create any more processes. It is not recommended to set this parameter to 0.

JOB_ACCEPT_INTERVAL set at the queue level (`lsb.queues`) overrides JOB_ACCEPT_INTERVAL set at the cluster level (`lsb.params`).

Default Undefined (the queue uses JOB_ACCEPT_INTERVAL defined in `lsb.params`, which has a default value of 1)

JOB_ACTION_WARNING_TIME

Syntax **JOB_ACTION_WARNING_TIME=**[*hour***:**]*minute*

Description Specifies the amount of time before a job control action occurs that a job warning action is to be taken. For example, 2 minutes before the job reaches run time limit or termination deadline, or the queue's run window is closed, an URG signal is sent to the job.

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.

Example JOB_ACTION_WARNING_TIME=2

Default Undefined

JOB_CONTROLS

Syntax **JOB_CONTROLS=SUSPEND**[*signal* | *command* | **CHKPNT**] **RESUME**[*signal* | *command*] **TERMINATE**[*signal* | *command* | **CHKPNT**]

- *signal* is a UNIX signal name (for example, SIGTSTP or SIGTERM). The specified signal is sent to the job.

  The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the `kill -l` command.
- *command* specifies a `/bin/sh` command line to be invoked.

  Do not quote the command line inside an action definition.

  Do not specify a signal followed by an action that triggers the same signal (for example, do not specify `JOB_CONTROLS=TERMINATE[bkill]` or `JOB_CONTROLS=TERMINATE[brequeue]`). This will cause a deadlock between the signal and the action.
- CHKPNT is a special action, which causes the system to checkpoint the job.

  Only valid for SUSPEND and TERMINATE actions:
  - If the SUSPEND action is CHKPNT, the job is checkpointed and then stopped by sending the SIGSTOP signal to the job automatically.
  - If the TERMINATE action is CHKPNT, then the job is checkpointed and killed automatically.

Description Changes the behavior of the SUSPEND, RESUME, and TERMINATE actions in LSF.

- The contents of the configuration line for the action are run with `/bin/sh -c` so you can use shell features in the command.

- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is `/dev/null`.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:
  - LSB_JOBPGIDS — a list of current process group IDs of the job
  - LSB_JOBPIDS —a list of current process IDs of the job

  For the SUSPEND action command, the following environment variables are also set:
  - LSB_SUSP_REASONS—an integer representing a bitmap of suspending reasons as defined in `lsbatch.h`

    The suspending reason can allow the command to take different actions based on the reason for suspending the job.
  - LSB_SUSP_SUBREASONS—an integer representing the load index that caused the job to be suspended

    When the suspending reason SUSP_LOAD_REASON (suspended by load) is set in LSB_SUSP_REASONS, LSB_SUSP_SUBREASONS set to one of the load index values defined in `lsf.h`.

  Use LSB_SUSP_REASONS and LSB_SUSP_SUBREASONS together in you custom job control to determine the exact load threshold that caused a job to be suspended.
- If an additional action is necessary for the SUSPEND command, that action should also send the appropriate signal to the application. Otherwise, a job can continue to run even after being suspended by LSF. For example,

  `JOB_CONTROLS=SUSPEND[bkill $LSB_JOBPIDS; `*`command`*`]`

Default On UNIX, by default, SUSPEND sends SIGTSTP for parallel or interactive jobs and SIGSTOP for other jobs. RESUME sends SIGCONT. TERMINATE sends SIGINT, SIGTERM and SIGKILL in that order.

On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications will be able to process them. Termination is implemented by the `TerminateProcess( )` system call.

JOB_IDLE

Syntax **`JOB_IDLE=`***number*

Description Specifies a threshold for idle job exception handling. The value should be a number between 0.0 and 1.0 representing CPU time/runtime. If the job idle factor is less than the specified threshold, LSF invokes `LSF_SERVERDIR/eadmin` to trigger the action for a job idle exception.

The minimum job run time before mbatchd reports that the job is idle is defined as `DETECT_IDLE_JOB_AFTER` in `lsb.params`.

Valid Values Any positive number between 0.0 and 1.0

Example `JOB_IDLE=0.10`

A job idle exception is triggered for jobs with an idle value (CPU time/runtime) less than 0.10.

Default Undefined. No job idle exceptions are detected.

JOB_OVERRUN

Syntax **JOB_OVERRUN=***run_time*

Description Specifies a threshold for job overrun exception handling. If a job runs longer than the specified run time, LSF invokes `LSF_SERVERDIR/eadmin` to trigger the action for a job overrun exception.

Example `JOB_OVERRUN=5`

A job overrun exception is triggered for jobs running longer than 5 minutes.

Default Undefined. No job overrun exceptions are detected.

JOB_STARTER

Syntax **JOB_STARTER=***starter* [*starter*] [**"%USRCMD"**] [*starter*]

Description Creates a specific environment for submitted jobs prior to execution.

starter is any executable that can be used to start the job (i.e., can accept the job as an input argument). Optionally, additional 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 job in the job starter command line. The %USRCMD string may be enclosed with quotes or followed by additional commands.

Example `JOB_STARTER=csh -c "%USRCMD;sleep 10"`

In this case, if a user submits a job

```
% bsub myjob arguments
```

the command that actually runs is:

```
% csh -c "myjob arguments;sleep 10"
```

Default Undefined (no job starter)

JOB_UNDERRUN

Syntax **JOB_UNDERRUN=***run_time*

Description Specifies a threshold for job underrun exception handling. If a job exits before the specified number of minutes, LSF invokes `LSF_SERVERDIR/eadmin` to trigger the action for a job underrun exception.

Example `JOB_UNDERRUN=2`

A job underrun exception is triggered for jobs running less than 2 minutes.

Default Undefined. No job underrun exceptions are detected.

JOB_WARNING_ACTION

Syntax **JOB_WARNING_ACTION=***signal*

Description Specifies the job action to be taken before a job control action occurs. For example, 2 minutes before the job reaches run time limit or termination deadline, or the queue's run window is closed, an URG signal is sent to the job.

A job warning action must be specified with a job action warning time in order for job warning to take effect.

If JOB_WARNING_ACTION 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.

You can specify actions similar to the JOB_CONTROLS queue level parameter: send a signal, invoke a command, or checkpoint the job.

The warning action specified by the `bsub -wa` option overrides JOB_WARNING_ACTION in the queue. JOB_WARNING_ACTION is used as the default when no command line option is specified.

Example JOB_WARNING_ACTION=URG

Default Undefined

load_index

Syntax *load_index*=*loadSched*[**/***loadStop*]

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index. Specify multiple lines to configure thresholds for multiple load indices.

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index as a column. Specify multiple columns to configure thresholds for multiple load indices.

Description Scheduling and suspending thresholds for the specified dynamic load index.

The `loadSched` condition must be satisfied before a job is dispatched to the host. If a RESUME_COND is not specified, the `loadSched` condition must also be satisfied before a suspended job can be resumed.

If the `loadStop` condition is satisfied, a job on the host will be suspended.

The `loadSched` and `loadStop` thresholds permit the specification of conditions using simple AND/OR logic. Any load index that does not have a configured threshold has no effect on job scheduling.

LSF will not suspend a job if the job is the only batch job running on the host and the machine is interactively idle (`it`>0).

The `r15s`, `r1m`, and `r15m` CPU run queue length conditions are compared to the effective queue length as reported by `lsload -E`, which is normalized for multiprocessor hosts. Thresholds for these parameters should be set at appropriate levels for single processor hosts.

Example

```
MEM=100/10
SWAP=200/30
```

These two lines translate into a `loadSched` condition of

```
mem>=100 && swap>=200
```

and a `loadStop` condition of

```
mem < 10 || swap < 30
```

Default Undefined

MANDATORY_EXTSCHED

Syntax **MANDATORY_EXTSCHED=***external_scheduler_options*

Description Specifies mandatory external scheduling options for the queue.

`-extsched` options on the `bsub` command are merged with MANDATORY_EXTSCHED options, and MANDATORY_EXTSCHED options override any conflicting job-level options set by `-extsched`.

Default Undefined

MAX_RSCHED_TIME

Syntax **MAX_RSCHED_TIME=***integer* | **infinit**

Description MultiCluster job forwarding model only. Determines how long a MultiCluster job stays pending in the execution cluster before returning to the submission cluster. The remote timeout limit in seconds is:

```
MAX_RSCHED_TIME * MBD_SLEEP_TIME=timeout
```

Specify `infinit` to disable remote timeout (jobs always get dispatched in the correct FCFS order because MultiCluster jobs never get rescheduled, but MultiCluster jobs can be pending in the receive-jobs queue forever instead of being rescheduled to a better queue).

Remote timeout limit never affects advance reservation jobs

Jobs that use an advance reservation always behave as if remote timeout is disabled.

Default 20 (20 minutes by default)

MEMLIMIT

Syntax **MEMLIMIT=**[*default_limit*] *maximum_limit*

Description The per-process (hard) process resident set size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit`(2)).

Sets the maximum amount of physical memory (resident set size, RSS) that may be allocated to a process.

By default, if a default memory limit is specified, jobs submitted to the queue without a job-level memory limit are killed when the default memory limit is reached.

If you specify only one limit, it is the maximum, or hard, memory limit. If you specify two limits, the first one is the default, or soft, memory limit, and the second one is the maximum memory limit.

LSF has two methods of enforcing memory usage:

- OS Memory Limit Enforcement
- LSF Memory Limit Enforcement

OS memory limit enforcement

OS memory limit enforcement is the default MEMLIMIT behavior and does not require further configuration. OS enforcement usually allows the process to eventually run to completion. LSF passes MEMLIMIT to the OS which uses it as a guide for the system scheduler and memory allocator. The system may allocate more memory to a process if there is a surplus. When memory is low, the system takes memory from and lowers the scheduling priority (re-nice) of a process that has exceeded its declared MEMLIMIT. Only available on systems that support `RLIMIT_RSS` for `setrlimit()`.

Not supported on:

- Sun Solaris 2.x
- Windows

LSF memory limit enforcement

To enable LSF memory limit enforcement, set LSB_MEMLIMIT_ENFORCE in `lsf.conf` to `y`. LSF memory limit enforcement explicitly sends a signal to kill a running process once it has allocated memory past MEMLIMIT.

You can also enable LSF memory limit enforcement by setting LSB_JOB_MEMLIMIT in `lsf.conf` to `y`. The difference between LSB_JOB_MEMLIMIT set to y and LSB_MEMLIMIT_ENFORCE set to y is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to y, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Available for all systems on which LSF collects total memory usage.

Example

The following configuration defines a queue with a memory limit of 5000 KB:

```
Begin Queue
QUEUE_NAME   = default
DESCRIPTION = Queue with memory limit of 5000 kbytes
MEMLIMIT     = 5000
End Queue
```

Default

Unlimited

MIG

Syntax

MIG=*minutes*

Description

Enables automatic job migration and specifies the migration threshold, in minutes.

Does not affect MultiCluster jobs that are forwarded to a remote cluster.

If a checkpointable or rerunnable job dispatched to the host is suspended (SSUSP state) for longer than the specified number of minutes, the job is migrated (unless another job on the same host is being migrated). A value of 0 (zero) specifies that a suspended job should be migrated immediately.

If a migration threshold is defined at both host and queue levels, the lower threshold is used.

Members of a chunk job can be migrated. Chunk jobs in WAIT state are removed from the job chunk and put into PEND state.

Default Undefined (no automatic job migration)

NEW_JOB_SCHED_DELAY

Syntax **NEW_JOB_SCHED_DELAY=***seconds*

Description The number of seconds that a new job waits, before being scheduled. A value of zero (0) means the job is scheduled without any delay.

Default 2 seconds

NICE

Syntax **NICE=***integer*

Description Adjusts the UNIX scheduling priority at which jobs from this queue execute.

The default value of 0 (zero) maintains the default scheduling priority for UNIX interactive jobs. This value adjusts the run-time priorities for batch jobs on a queue-by-queue basis, to control their effect on other batch or interactive jobs. See the `nice`(1) manual page for more details.

On Windows, this value is mapped to Windows process priority classes as follows:

- `nice>=0` corresponds to an priority class of `IDLE`
- `nice<0` corresponds to an priority class of `NORMAL`

Platform LSF on Windows does not support `HIGH` or `REAL-TIME` priority classes.

Default 0 (zero)

NQS_QUEUES

Syntax **NQS_QUEUES=***NQS_ queue_name***@***NQS_host_name* ...

Description Makes the queue an NQS forward queue.

NQS_host_name is an NQS host name that can be the official host name or an alias name known to the LSF master host through `gethostbyname`(3).

NQS_queue_name is the name of an NQS destination queue on this host. NQS destination queues are considered for job routing in the order in which they are listed here. If a queue accepts the job, it is routed to that queue. If no queue accepts the job, it remains pending in the NQS forward queue.

`lsb.nqsmaps` must be present for the LSF system to route jobs in this queue to NQS systems.

You must configure LSB_MAX_NQS_QUEUES in `lsf.conf` to specify the maximum number of NQS queues allowed in the LSF cluster. This is required for LSF to work with NQS.

Since many features of LSF are not supported by NQS, the following queue configuration parameters are ignored for NQS forward queues: PJOB_LIMIT, POLICIES, RUN_WINDOW, DISPATCH_WINDOW, RUNLIMIT, HOSTS, MIG. In addition, scheduling load threshold parameters are ignored because NQS does not provide load information about hosts.

Default Undefined

PJOB_LIMIT

Syntax **PJOB_LIMIT=***float*

Description Per-processor job slot limit for the queue.

Maximum number of job slots that this queue can use on any processor. This limit is configured per processor, so that multiprocessor hosts automatically run more jobs.

Default Unlimited

POST_EXEC

Syntax **POST_EXEC=***command*

Description A command run on the execution host after the job.

UNIX The entire contents of the configuration line of the pre- and post-execution commands are run under `/bin/sh -c`, so shell features can be used in the command.

The pre- and post-execution commands are run in `/tmp`.

Standard input and standard output and error are set to:

`/dev/null`

The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.

The PATH environment variable is set to:

`'/bin /usr/bin /sbin/usr/sbin'`

Windows The pre- and post-execution commands are run under `cmd.exe/c`.

To run these commands under a different user account (such as root, to do privileged operations, if necessary), configure the parameter LSB_PRE_POST_EXEC_USER in `lsf.sudoers`.

Standard input and standard output and error are set to NUL. The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.

The PATH is determined by the setup of the LSF Service.

- Other environment variables set for the job are also set for the pre- and post-execution commands.
- When a job is dispatched from a queue that has a pre-execution command, the system will remember the post-execution command defined for the queue from which the job is dispatched. If the job is later switched to another queue or the post-execution command of the queue is changed, the original post-execution command will be run.

- When the post-execution command is run, the environment variable LSB_JOBEXIT_STAT is set to the exit status of the job. Refer to the manual page for `wait`(2) for the format of this exit status.
- The post-execution command is also run if a job is requeued because the job's execution environment fails to be set up or if the job exits with one of the queue's REQUEUE_EXIT_VALUES. The environment variable LSB_JOBPEND is set if the job is requeued. If the job's execution environment could not be set up, LSB_JOBEXIT_STAT is set to 0 (zero).

Default No post-execution commands

PRE_EXEC

Syntax **`PRE_EXEC=`***command*

Description A command run on the execution host before the job.

To specify a pre-execution command at the job level, use `bsub -E`. If both queue and job level pre-execution commands are specified, the job level pre-execution is run after the queue level pre-execution command.

If the pre-execution command exits with a non-zero exit code, it is considered to have failed, and the job is requeued to the head of the queue. This feature can be used to implement customized scheduling by having the pre-execution command fail if conditions for dispatching the job are not met.

Other environment variables set for the job are also set for the pre- and post-execution commands.

UNIX The entire contents of the configuration line of the pre- and post-execution commands are run under `/bin/sh -c`, so shell features can be used in the command.

The pre- and post-execution commands are run in `/tmp`.

Standard input and standard output and error are set to: `/dev/null`

The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.

The PATH environment variable is set to: `/bin /usr/bin /sbin/usr/sbin`

Windows The pre- and post-execution commands are run under `cmd.exe/c`.

To run these commands under a different user account (such as root, to do privileged operations, if necessary), configure the parameter LSB_PRE_POST_EXEC_USER in `lsf.sudoers`.

Standard input and standard output and error are set to NUL. The output from the pre- and post-execution commands can be explicitly redirected to a file for debugging purposes.

The PATH is determined by the setup of the LSF Service.

Default No pre-execution commands

PREEMPTION

Syntax **`PREEMPTION=PREEMPTIVE`**`[`**`[`***`queue_name`*`[`**`+`***`pref_level`*`]...`**`]`**`]`
`PREEMPTABLE``[`**`[`***`queue_name`*`...`**`]`**`]`

Description Enables preemptive scheduling and defines a preemption policy for the queue.

You can specify PREEMPTIVE or PREEMPTABLE or both.When you specify a list of queues, you must enclose the list in one set of square brackets.

- PREEMPTIVE defines a preemptive queue. Jobs in this queue preempt jobs from the specified lower-priority queues or from all lower-priority queues by default (if the parameter is specified with no queue names).

  If you specify a list of lower-priority queues, you must enclose the list in one set of square brackets. To indicate an order of preference for the lower-priority queues, put a plus sign (+) after the names of queues and a preference level as a positive integer.
- PREEMPTABLE defines a preemptable queue. Jobs in this queue can be preempted by jobs from specified higher-priority queues, or from all higher-priority queues by default, even if the higher-priority queues are not preemptive. If you specify a list of higher-priority queues, you must enclose the list in one set of square brackets.

PREEMPTIVE and PREEMPTABLE can be used together, to specify that jobs in this queue can always preempt jobs in lower-priority queues and can always be preempted by jobs from higher-priority queues.

PRIORITY

Syntax `PRIORITY=`*integer*

Description The queue priority. A higher value indicates a higher LSF dispatching priority, relative to other queues.

LSF schedules jobs from one queue at a time, starting with the highest-priority queue. If multiple queues have the same priority, LSF schedules all the jobs from these queues in first-come, first-served order.

However, only jobs from FCFS queues are scheduled together. If fairshare queues have the same priority, the jobs are always scheduled queue-by-queue, in the order in which the queues are listed in `lsb.queues`. If a cluster has both FCFS and fairshare queues all having the same priority, the `lsb.queues` order is considered, but all the FCFS jobs are scheduled at once, when the first FCFS queue has its turn.

Queue priority in LSF is completely independent of the UNIX scheduler's priority system for time-sharing processes. In LSF, the NICE parameter is used to set the UNIX time-sharing priority for batch jobs.

Default 1 (lowest possible priority)

PROCESSLIMIT

Syntax `PROCESSLIMIT=`[*default_limit*] *maximum_limit*

Description Limits the number of concurrent processes that can be part of a job.

By default, if a default process limit is specified, jobs submitted to the queue without a job-level process limit are killed when the default process limit is reached.

If you specify only one limit, it is the maximum, or hard, process limit. If you specify two limits, the first one is the default, or soft, process limit, and the second one is the maximum process limit.

Default Unlimited

PROCLIMIT

Syntax **PROCLIMIT=**[*minimum_limit* [*default_limit*]] *maximum_limit*

Description Maximum number of slots that can be allocated to a job. For parallel jobs, the maximum number of processors that can be allocated to the job.

Optionally specifies the minimum and default number of job slots.

All limits must be positive numbers greater than or equal to 1 that satisfy the following relationship:

1 <= *minimum* <= *default* <= *maximum*

You can specify up to three limits in the PROCLIMIT parameter:

If You Specify ...	Then ...
One limit	It is the maximum processor limit. The minimum and default limits are set to 1.
Two limits	The first is the minimum processor limit, and the second one is the maximum. The default is set equal to the minimum. The minimum must be less than or equal to the maximum.
Three limits	The first is the minimum processor limit, the second is the default processor limit, and the third is the maximum. The minimum must be less than the default and the maximum.

Jobs that request fewer slots than the minimum PROCLIMIT or more slots than the maximum PROCLIMIT cannot use the queue and are rejected. If the job requests minimum and maximum job slots, the maximum slots requested cannot be less than the minimum PROCLIMIT, and the minimum slots requested cannot be more than the maximum PROCLIMIT.

Default Unlimited, the default number of slots is 1

QJOB_LIMIT

Syntax **QJOB_LIMIT=***integer*

Description Job slot limit for the queue. Total number of job slots that this queue can use.

Default Unlimited

QUEUE_NAME

Syntax **QUEUE_NAME=***string*

Description Required. Name of the queue.

Specify any ASCII string up to 60 characters long. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces. You cannot specify the reserved name `default`.

Default You must specify this parameter to define a queue. The default queue automatically created by LSF is named `default`.

RCVJOBS_FROM

Syntax **RCVJOBS_FROM=***cluster_name* ... | **allclusters**

Description MultiCluster only. Defines a MultiCluster receive-jobs queue.

Specify cluster names, separated by a space. The administrator of each remote cluster determines which queues in that cluster will forward jobs to the local cluster.

Use the keyword `allclusters` to specify any remote cluster.

Example `RCVJOBS_FROM=cluster2 cluster4 cluster6`

This queue accepts remote jobs from clusters 2, 4, and 6.

REQUEUE_EXIT_VALUES

Syntax **REQUEUE_EXIT_VALUES=**[*exit_code* ...] [**EXCLUDE(***exit_code* ...**)**]

Description Enables automatic job requeue and sets the LSB_EXIT_REQUEUE environment variable. Separate multiple exit codes with spaces.

Jobs are requeued to the head of the queue. The output from the failed run is not saved, and the user is not notified by LSF.

Define an exit code as EXCLUDE(*exit_code*) to enable exclusive job requeue. Exclusive job requeue does not work for parallel jobs.

For MultiCluster jobs forwarded to a remote execution cluster, the exit values specified in the submission cluster with the EXCLUSIVE keyword are treated as if they were non-exclusive.

If `mbatchd` is restarted, it will 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.

You should configure REQUEUE_EXIT_VALUES for interruptible backfill queues (INTERRUPTIBLE_BACKFILL=*seconds*).

Example `REQUEUE_EXIT_VALUES=30 EXCLUDE(20)`

means that jobs with exit code 30 are requeued, jobs with exit code 20 are requeued exclusively, and jobs with any other exit code are not requeued.

Default Undefined (jobs in this queue are not requeued)

RERUNNABLE

Syntax **RERUNNABLE=yes** | **no**

Description If `yes`, enables automatic job rerun (restart).

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.

Default `no`

RESOURCE_RESERVE

Syntax **RESOURCE_RESERVE=MAX_RESERVE_TIME[***integer***]**

Description Enables processor reservation and memory reservation for pending jobs for the queue. Specifies the number of dispatch turns (MAX_RESERVE_TIME) over which a job can reserve job slots and memory.

Overrides the SLOT_RESERVE parameter. If both RESOURCE_RESERVE and SLOT_RESERVE are defined in the same queue, an error is displayed when the cluster is reconfigured, and SLOT_RESERVE is ignored. Job slot reservation for parallel jobs is enabled by RESOURCE_RESERVE if the LSF scheduler plugin module names for both resource reservation and parallel batch jobs (`schmod_parallel` and `schmod_reserve`) are configured in the `lsb.modules` file: The `schmod_parallel` name *must* come before `schmod_reserve` in `lsb.modules`.

If a job has not accumulated enough memory or job slots to start by the time MAX_RESERVE_TIME expires, it releases all its reserved job slots or memory so that other pending jobs can run. After the reservation time expires, the job cannot reserve memory or slots for one scheduling session, so other jobs have a chance to be dispatched. After one scheduling session, the job can reserve available memory and job slots again for another period specified by MAX_RESERVE_TIME.

If BACKFILL is configured in a queue, and a run limit is specified with `-W` on `bsub` or with RUNLIMIT in the queue, backfill jobs can use the accumulated memory reserved by the other jobs in the queue, as long as the backfill job can finish before the predicted start time of the jobs with the reservation.

Unlike slot reservation, which only applies to parallel jobs, memory reservation and backfill on memory apply to sequential and parallel jobs.

Example `RESOURCE_RESERVE=MAX_RESERVE_TIME[5]`

This example specifies that jobs have up to 5 dispatch turns to reserve sufficient job slots or memory (equal to 5 minutes, by default).

Default Undefined (no job slots or memory reserved)

RES_REQ

Syntax **RES_REQ=***res_req*

Description Resource requirements used to determine eligible hosts. Specify a resource requirement string as usual. The resource requirement string lets you specify conditions in a more flexible manner than using the load thresholds.

The `select` section defined at the queue level must be satisfied at in addition to any job-level requirements or load thresholds.

The `rusage` section can specify additional requests. To do this, use the `OR` (||) operator to separate additional `rusage` strings.

When both job-level and queue-level `rusage` sections are defined, the `rusage` section defined for the job overrides the `rusage` section defined in the queue. The two `rusage` definitions are merged, with the job-level `rusage` taking precedence. For example:

- Given a RES_REQ definition in a queue:

  `RES_REQ=rusage[mem=200:lic=1] ...`

  and job submission:

  `bsub -R'rusage[mem=100]' ...`

  The resulting requirement for the job is

  `rusage[mem=100:lic=1]`

  where `mem=100` specified by the job overrides `mem=200` specified by the queue. However, `lic=1` from queue is kept, since job does not specify it.
- For the following queue-level RES_REQ (decay and duration defined):

  `RES_REQ=rusage[mem=200:duration=20:decay=1] ...`

  and job submission (no decay or duration):

  `bsub -R'rusage[mem=100]' ...`

  The resulting requirement for the job is:

  `rusage[mem=100:duration=20:decay=1]`

  Queue-level duration and decay are merged with the job-level specification, and `mem=100` for the job overrides `mem=200` specified by the queue. However, `duration=20` and `decay=1` from queue are kept, since job does not specify them.

The `order` section defined at the queue level is ignored if any resource requirements are specified at the job level (if the job-level resource requirements do not include the `order` section, the default order, `r15s:pg`, is used instead of the queue-level resource requirement).

The `span` section defined at the queue level is ignored if the `span` section is also defined at the job level.

If RES_REQ is defined at the queue level and there are no load thresholds defined, the pending reasons for each individual load index will not be displayed by `bjobs`.

Default

`select[type==local] order[r15s:pg]`. If this parameter is defined and a host model or Boolean resource is specified, the default type will be `any`.

RESUME_COND

Syntax

RESUME_COND=*res_req*

Use the `select` section of the resource requirement string to specify load thresholds. All other sections are ignored.

Description

LSF automatically resumes a suspended (SSUSP) job in this queue if the load on the host satisfies the specified conditions.

If RESUME_COND is not defined, then the `loadSched` thresholds are used to control resuming of jobs. The `loadSched` thresholds are ignored, when resuming jobs, if RESUME_COND is defined.

RUN_WINDOW

Syntax

RUN_WINDOW=*time_window* ...

Description Time periods during which jobs in the queue are allowed to run.

When the window closes, LSF suspends jobs running in the queue and stops dispatching jobs from the queue. When the window reopens, LSF resumes the suspended jobs and begins dispatching additional jobs.

Default Undefined (queue is always active)

RUNLIMIT

Syntax **RUNLIMIT=**[*default_limit*] *maximum_limit*

where *default_limit* and *maximum_limit* are:

[*hour*:]*minute*[/*host_name* | /*host_model*]

Description The maximum run limit and optionally the default run limit. The name of a host or host model specifies the run time normalization host to use.

By default, jobs that are in the RUN state for longer than the specified maximum run limit are killed by LSF. You can optionally provide your own termination job action to override this default.

Jobs submitted with a job-level run limit (`bsub -W`) that is less than the maximum run limit are killed when their job-level run limit is reached. Jobs submitted with a run limit greater than the maximum run limit are rejected by the queue.

If a default run limit is specified, jobs submitted to the queue without a job-level run limit are killed when the default run limit is reached. The default run limit is used with backfill scheduling of parallel jobs.

If you specify only one limit, it is the maximum, or hard, run limit. If you specify two limits, the first one is the default, or soft, run limit, and the second one is the maximum run limit. The number of minutes may be greater than 59. Therefore, three and a half hours can be specified either as 3:30, or 210.

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 run time limit is not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted to a queue with a run limit configured.

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. (See `lsinfo`(1) to get host model information.)

If no host or host model is given, LSF uses the default run time 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, the host with the largest CPU factor (the fastest host in the cluster).

For MultiCluster 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).

Jobs submitted to a chunk job queue are not chunked if RUNLIMIT is greater than 30 minutes.

RUNLIMIT is required for queues configured with INTERRUPTIBLE_BACKFILL.

Default Unlimited

SLOT_POOL

Syntax **`SLOT_POOL=`***pool_name*

Description Name of the pool of job slots the queue belongs to for queue-based fairshare. A queue can only belong to one pool. All queues in the pool must share the same set of hosts.

Valid value Specify any ASCII string up to 60 characters long. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

Default Undefined (no job slots reserved)

SLOT_RESERVE

Syntax **`SLOT_RESERVE=MAX_RESERVE_TIME[`***integer***`]`**

Description Enables processor reservation for the queue and specifies the reservation time. Specify the keyword MAX_RESERVE_TIME and, in square brackets, the number of MBD_SLEEP_TIME cycles over which a job can reserve job slots. MBD_SLEEP_TIME is defined in `lsb.params`; the default value is 60 seconds.

If a job has not accumulated enough job slots to start before the reservation expires, it releases all its reserved job slots so that other jobs can run. Then, the job cannot reserve slots for one scheduling session, so other jobs have a chance to be dispatched. After one scheduling session, the job can reserve job slots again for another period specified by SLOT_RESERVE.

SLOT_RESERVE is overridden by the RESOURCE_RESERVE parameter.

If both RESOURCE_RESERVE and SLOT_RESERVE are defined in the same queue, job slot reservation and memory reservation are enabled and an error is displayed when the cluster is reconfigured. SLOT_RESERVE is ignored.

Job slot reservation for parallel jobs is enabled by RESOURCE_RESERVE if the LSF scheduler plugin module names for both resource reservation and parallel batch jobs (`schmod_parallel` and `schmod_reserve`) are configured in the `lsb.modules` file: The `schmod_parallel` name *must* come before `schmod_reserve` in `lsb.modules`.

If BACKFILL is configured in a queue, and a run limit is specified with `-W` on `bsub` or with RUNLIMIT in the queue, backfill parallel jobs can use job slots reserved by the other jobs, as long as the backfill job can finish before the predicted start time of the jobs with the reservation.

Unlike memory reservation, which applies both to sequential and parallel jobs, slot reservation applies only to parallel jobs.

Example `SLOT_RESERVE=MAX_RESERVE_TIME[5]`

This example specifies that parallel jobs have up to 5 cycles of MBD_SLEEP_TIME (5 minutes, by default) to reserve sufficient job slots to start.

Default Undefined (no job slots reserved)

SLOT_SHARE

Syntax **SLOT_SHARE=***integer*

Description Share of job slots for queue-based fairshare. Represents the percentage of running jobs (job slots) in use from the queue. SLOT_SHARE must be greater than zero (0) and less than or equal to 100.

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.

Default Undefined

SNDJOBS_TO

Syntax **SNDJOBS_TO=***queue_name***@***cluster_name* ...

Description Defines a MultiCluster send-jobs queue.

Specify remote queue names, in the form *queue_name***@***cluster_name*, separated by a space.

This parameter is ignored if `lsb.queues` HOSTS specifies remote (borrowed) resources.

Example `SNDJOBS_TO=queue2@cluster2 queue3@cluster2 queue3@cluster3`

STACKLIMIT

Syntax **STACKLIMIT=***integer*

Description The per-process (hard) stack segment size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit`(2)).

Default Unlimited

STOP_COND

Syntax **STOP_COND=***res_req*

Use the `select` section of the resource requirement string to specify load thresholds. All other sections are ignored.

Description LSF automatically suspends a running job in this queue if the load on the host satisfies the specified conditions.

- LSF will not suspend the only job running on the host if the machine is interactively idle (`it` > 0).
- LSF will not suspend a forced job (`brun -f`).
- LSF will not suspend a job because of paging rate if the machine is interactively idle.

If STOP_COND is specified in the queue and there are no load thresholds, the suspending reasons for each individual load index will not be displayed by `bjobs`.

Example

```
STOP_COND= select[((!cs && it < 5) || (cs && mem < 15 && swp
< 50))]
```

In this example, assume "cs" is a Boolean resource indicating that the host is a computer server. The stop condition for jobs running on computer servers is based on the availability of swap memory. The stop condition for jobs running on other kinds of hosts is based on the idle time.

SWAPLIMIT

Syntax

SWAPLIMIT=*integer*

Description

The amount of total virtual memory limit (in KB) for a job from this queue.

This limit applies to the whole job, no matter how many processes the job may contain.

The action taken when a job exceeds its SWAPLIMIT or PROCESSLIMIT is to send SIGQUIT, SIGINT, SIGTERM, and SIGKILL in sequence. For CPULIMIT, SIGXCPU is sent before SIGINT, SIGTERM, and SIGKILL.

Default

Unlimited

TERMINATE_WHEN

Syntax

TERMINATE_WHEN=[**LOAD**] [**PREEMPT**] [**WINDOW**]

Description

Configures the queue to invoke the TERMINATE action instead of the SUSPEND action in the specified circumstance.

- LOAD — kills jobs when the load exceeds the suspending thresholds.
- PREEMPT — kills jobs that are being preempted.
- WINDOW — kills jobs if the run window closes.

If the TERMINATE_WHEN job control action is applied to a chunk job, `sbatchd` kills the chunk job element that is running and puts the rest of the waiting elements into pending state to be rescheduled later.

Example

Set TERMINATE_WHEN to WINDOW to define a night queue that will kill jobs if the run window closes:

```
Begin Queue
NAME            = night
RUN_WINDOW      = 20:00-08:00
TERMINATE_WHEN = WINDOW
JOB_CONTROLS    = TERMINATE[kill -KILL $LS_JOBPGIDS; mail - s
"job $LSB_JOBID killed by queue run window" $USER < /dev/null]
End Queue
```

THREADLIMIT

Syntax **THREADLIMIT=**[*default_limit*] *maximum_limit*

Description Limits the number of concurrent threads that can be part of a job. 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.

By default, if a default thread limit is specified, jobs submitted to the queue without a job-level thread limit are killed when the default thread limit is reached.

If you specify only one limit, it is the maximum, or hard, thread limit. If you specify two limits, the first one is the default, or soft, thread limit, and the second one is the maximum thread limit.

Both the default and the maximum limits must be positive integers. The default limit must be less than the maximum limit. The default limit is ignored if it is greater than the maximum limit.

Examples `THREADLIMIT=6`

No default thread limit is specified. The value 6 is the default and maximum thread limit.

`THREADLIMIT=6 8`

The first value (6) is the default thread limit. The second value (8) is the maximum thread limit.

Default Unlimited

UJOB_LIMIT

Syntax **UJOB_LIMIT**=*integer*

Description Per-user job slot limit for the queue. Maximum number of job slots that each user can use in this queue.

Default Unlimited

USERS

Syntax **USERS=all** [**~***user_name* ...] [**~***user_group* ...] | [*user_name* ...] [*user_group* [**~***user_group* ...] ...]

Description A space-separated list of user names or user groups that can submit jobs to the queue. Use the reserved word `all` to specify all LSF users. LSF cluster administrators are automatically included in the list of users, so LSF cluster administrators can submit jobs to this queue, or switch any user's jobs into this queue, even if they are not listed.

If user groups are specified, each user in the group can submit jobs to this queue. If FAIRSHARE is also defined in this queue, only users defined by both parameters can submit jobs, so LSF administrators cannot use the queue if they are not included in the share assignments.

User names must be valid login names.

User group names can be LSF user groups or UNIX and Windows user groups.

Use the keyword `all` to specify all users or user groups in a cluster.

Use the not operator (~) to exclude users from the `all` specification or from user groups. This is useful if you have a large number of users but only want to exclude a few users or groups from the queue definition.

The not operator can only be used with the `all` keyword or to exclude users from user groups.

The not operator does not exclude LSF administrators from the queue definintion.

Default

`all` (all users can submit jobs to the queue)

Examples

- `USERS=user1 user2`
- `USERS=all ~user1 ~user2`
- `USERS=all ~ugroup1`
- `USERS=groupA ~user3 ~user4`

Automatic Time-based Configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.queues` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the `badmin reconfig` command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

```
Begin Queue
...
#if time(8:30-18:30)
   INTERACTIVE  = ONLY  # interactive only during day shift
#endif
...
End Queue
```

SEE ALSO

lsf.cluster(5), lsf.conf(5), lsb.params(5), lsb.hosts(5), lsb.users(5), lsf.sudoers(5), bhpart(1), busers(1), bchkpnt(1), bugroup(1), bmgroup(1), nice(1), getgrnam(3), getrlimit(2), bqueues(1), bhosts(1), bsub(1), lsid(1), mbatchd(8), badmin(8)

lsb.resources

The lsb.resources file contains configuration information for resource allocation limits, exports, and resource usage limits. This file is optional.

The lsb.resources file is stored in the directory LSB_CONFDIR/*cluster_name*/configdir, where LSB_CONFDIR is defined in lsf.conf.

Changing lsb.resources configuration

After making any changes to lsb.resources, run badmin reconfig to reconfigure mbatchd.

Contents

- "Limit Section" on page 434
- "HostExport Section" on page 447
- "SharedResourceExport Section" on page 450
- "ResourceReservation Section" on page 451
- "ReservationUsage Section" on page 454
- "Automatic Time-based Configuration" on page 455

Limit Section

Sets limits for the maximum amount of the specified resources must be available for different classes of jobs to start, and which resource consumers the limits apply to. Limits are enforced during job resource allocation.

For limits to be enforced, jobs must specify `rusage` resource requirements (`bsub -R` or RES_REQ in `lsb.queues`).

The `blimits` command displays view current usage of resource allocation limit configured in `Limit` sections in `lsb.resources`:

Limit section structure

Each set of limits is defined in a Limit section enclosed by `Begin Limit` and `End Limit`. Limit sections set limits for how much resources must be available for different classes of jobs to start.

A Limit section has two formats:

- Vertical tabular
- Horizontal

The file can contain sections in both formats. In either format, you must configure a limit for at least one consumer and one resource. The Limit section cannot be empty.

Vertical tabular format

Use the vertical format for simple configuration conditions involving only a few consumers and resource limits.

The first row consists of the following keywords for:

- Resource types:
  - SLOTS or SLOTS_PER_PROCESSOR
  - MEM (MB or percentage)
  - SWP (MB or percentage)
  - TMP (MB or percentage)
  - LICENSE
  - RESOURCE
- Consumer types:
  - USERS or PER_USER
  - QUEUES or PER_QUEUE
  - HOSTS or PER_HOST
  - PROJECTS or PER_PROJECT

Each subsequent row describes the configuration information for resource consumers and the limits that apply to them. Each line must contain an entry for each keyword. Use empty parentheses `()` or a dash (`-`) to specify the default value for an entry. Fields cannot be left blank. For resources, the default is no limit; for consumers, the default is all consumers.

Multiple entries must be enclosed in parentheses. For RESOURCE and LICENSE limits, resource and license names must be enclosed in parentheses.

Horizontal format

Use the horizontal format to give a name for your limits and to configure more complicated combinations of consumers and resource limits.

The first line of the Limit section gives the name of the limit configuration.

Each subsequent line in the Limit section consists of keywords identifying the resource limits:

- Job slots and per-processor job slots
- Memory (MB or percentage)
- Swap space (MB or percentage)
- Tmp space (MB or percentage)
- Software licenses
- Other shared resources

and the resource *consumers* to which the limits apply:

- Users and user groups
- Hosts and host groups
- Queues
- Projects

Example Limit sections

Vertical tabular format

In the following limit configuration, `user1` and `user3` are limited to 2 job slots on `hostA`, and jobs from `user2` on queue `normal` are limited to 20 MB of memory:

```
Begin Limit
USERS          QUEUES        HOSTS     SLOTS  MEM   SWP  TMP
(user1 user3)  -             hostA     2      -      -    -
user2          normal        -         -      20     -    -
End Limit
```

Jobs that do not match these limits; that is, all users except `user1` and `user3` running jobs on `hostA` and all users except `user2` submitting jobs to queue `normal`, have no limits.

Horizontal format

All users in user group `ugroup1` except `user1` using `queue1` and `queue2` and running jobs on hosts in host group `hgroup1` are limited to 2 job slots per processor on each host:

```
Begin Limit
# ugroup1 except user1 uses queue1 and queue2 with 2 job slots
# on each host in hgroup1
NAME           = limit1
# Resources
SLOTS_PER_PROCESSOR = 2
#Consumers
QUEUES        = queue1 queue2
USERS         = ugroup1 ~user1
PER_HOST      = hgroup1
End Limit
```

Compatibility with lsb.queues, lsb.users, and lsb.hosts

The Limit section of `lsb.resources` does not support the keywords or format used in `lsb.users`, `lsb.hosts`, and `lsb.queues`. However, your existing job slot limit configuration in these files will continue to apply.

Job slot limits are the only type of limit you can configure in `lsb.users`, `lsb.hosts`, and `lsb.queues`. You cannot configure limits for user groups, host groups, and projects in `lsb.users`, `lsb.hosts`, and `lsb.queues`. You should not configure any new resource allocation limits in `lsb.users`, `lsb.hosts`, and `lsb.queues`. Use `lsb.resources` to configure all new resource allocation limits, including job slot limits.

Existing limits in `lsb.users`, `lsb.hosts`, and `lsb.queues` with the same scope as a new limit in `lsb.resources`, but with a different value are ignored. The value of the new limit in `lsb.resources` is used. Similar limits with different scope enforce the most restrictive limit.

HOSTS

Syntax

HOSTS=all [**~**]*host_name* ... | **all** [**~**]*host_group* ...

HOSTS
([**-**] | **all** [**~**]*host_name* ... | **all** [**~**]*host_group* ... **)**

Description

A space-separated list of hosts, host groups defined in `lsb.hosts` on which limits are enforced. Limits are enforced on all hosts or host groups listed.

If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

To specify a per-host limit, use the PER_HOST keyword. Do not configure HOSTS and PER_HOST limits in the same Limit section.

If you specify MEM, TMP, or SWP as a percentage, you must specify PER_HOST and list the hosts that the limit is to be enforced on. You cannot specify HOSTS.

In horizontal format, use only one HOSTS line per Limit section.

Use the keyword `all` to configure limits that apply to all hosts in a cluster.

Use the not operator (~) to exclude hosts from the `all` specification in the limit. This is useful if you have a large cluster but only want to exclude a few hosts from the limit definition.

In vertical tabular format, multiple host names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate all hosts. Fields cannot be left blank.

Default

`all` (limits are enforced on all hosts in the cluster).

Example 1

`HOSTS=Group1 ~hostA hostB hostC`

Enforces limits on `hostB`, `hostC`, and all hosts in `Group1` except for `hostA`.

Example 2

`HOSTS=all ~group2 ~hostA`

Enforces limits on all hosts in the cluster, except for `hostA` and the hosts in `group2`.

Example 3

```
HOSTS                         SWP
(all ~hostK ~hostM)            10
```

Enforces a 10 MB swap limit on all hosts in the cluster, except for `hostK` and `hostM`

LICENSE

Syntax

LICENSE=[*license_name***,***integer***]** [**[***license_name***,***integer***]** ...]

LICENSE
(**[***license_name***,***integer***]** [**[***license_name***,***integer***]** ...] **)**

Description

Maximum number of specified software licenses available to resource consumers. The value must be a positive integer greater than or equal to zero.

Software licenses must be defined as decreasing numeric shared resources in `lsf.shared`.

The RESOURCE keyword is a synonym for the LICENSE keyword. You cannot specify RESOURCE and LICENSE in the same Limit section.

In horizontal format, use only one LICENSE line per Limit section.

In vertical tabular format, license entries must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate the default value (no limit). Fields cannot be left blank.

Default

None

Examples

```
LICENSE=[verilog,4] [spice,2]
```

```
Begin Limit
LICENSE                       PER_HOST
([verilog, 1])                (all ~hostA)
([verilog, 1] [spice,2])      (hostA)
End Limit
```

MEM

Syntax

MEM=*integer*[**%**]

MEM
- | *integer*[**%**]

Description

Maximum amount of memory available to resource consumers. Specify a value in MB or a percentage (%) as a positive integer greater than or equal 0. If you specify a percentage, you must also specify PER_HOST and list the hosts that the limit is to be enforced on.

The Limit section is ignored if MEM is specified as a percentage:

- Without PER_HOST

OR

- With HOSTS

In horizontal format, use only one MEM line per Limit section.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate the default value (no limit). Fields cannot be left blank.

If only QUEUES are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory available to the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory that the users or user groups can use on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, MEM must be an integer value. It cannot be a percentage. MEM is the maximum amount of memory available to the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit memory available to resource consumers.

Default No limit

Example `MEM=20`

NAME

Syntax **NAME=***text*

Description Required. Name of the Limit section

Specify any ASCII string 40 characters or less. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

Format Horizontal only

Default None. You must provide a name for the Limit section.

Example `NAME=short_limits`

PER_HOST

Syntax **PER_HOST=all** [~]*host_name* ... | **all** [~]*host_group* ...

PER_HOST
([**-**] | **all** [~]*host_name* ... | **all** [~]*host_group* ... **)**

Description A space-separated list of host or host groups defined in `lsb.hosts` on which limits are enforced. Limits are enforced on each host or individually to each host of the host group listed. If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

Do not configure PER_HOST and HOSTS limits in the same Limit section.

In horizontal format, use only one PER_HOST line per Limit section.

If you specify MEM, TMP, or SWP as a percentage, you must specify PER_HOST and list the hosts that the limit is to be enforced on. You cannot specify HOSTS.

Use the keyword `all` to configure limits that apply to each host in a cluster. If host groups are configured, the limit applies to each member of the host group, not the group as a whole.

Use the not operator (~) to exclude hosts or host groups from the `all` specification in the limit. This is useful if you have a large cluster but only want to exclude a few hosts from the limit definition.

In vertical tabular format, multiple host names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate each host or host group member. Fields cannot be left blank.

Default None. If no limit is specified for PER_HOST or HOST, no limit is enforced on any host or host group.

Example `PER_HOST=hostA hgroup1 ~hostC`

PER_PROJECT

Syntax **PER_PROJECT=all** [**~**]*project_name* ...

PER_PROJECT
([**-**] | **all** [**~**]*project_name* ...**)**

Description A space-separated list of project names on which limits are enforced. Limits are enforced on each project listed.

Do not configure PER_PROJECT and PROJECTS limits in the same Limit section.

In horizontal format, use only one PER_PROJECT line per Limit section.

Use the keyword `all` to configure limits that apply to each project in a cluster.

Use the not operator (~) to exclude projects from the `all` specification in the limit.

In vertical tabular format, multiple project names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate each project. Fields cannot be left blank.

Default None. If no limit is specified for PER_PROJECT or PROJECTS, no limit is enforced on any project.

Example `PER_PROJECT=proj1 proj2`

PER_QUEUE

Syntax **PER_QUEUE=all** [**~**]*queue_name* ..

PER_QUEUES
([**-**] | **all** [**~**]*queue_name* ...**)**

Description A space-separated list of queue names on which limits are enforced. Limits are enforced on jobs submitted to each queue listed.

Do not configure PER_QUEUE and QUEUES limits in the same Limit section.

In horizontal format, use only one PER_QUEUE line per Limit section.

Use the keyword `all` to configure limits that apply to each queue in a cluster.

Use the not operator (~) to exclude queues from the `all` specification in the limit. This is useful if you have a large number of queues but only want to exclude a few queues from the limit definition.

In vertical tabular format, multiple queue names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate each queue. Fields cannot be left blank.

Default None. If no limit is specified for PER_QUEUE or QUEUES, no limit is enforced on any queue.

Example `PER_QUEUE=priority night`

PER_USER

Syntax **PER_USER=all** [**~**]*user_name* ... | **all** [**~**]*user_group* ...

PER_USER
([**-**] | **all** [**~**]*user_name* ... | **all** [**~**]*user_group* ... **)**

Description A space-separated list of user names or user groups on which limits are enforced. Limits are enforced on each user or individually to each user in the user group listed. If a user group contains a subgroup, the limit also applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups. Note that for LSF and UNIX user groups, the groups must be specified in `lsb.users` first. See “UserGroup Section” on page 466.

Do not configure PER_USER and USERS limits in the same Limit section.

In horizontal format, use only one PER_USER line per Limit section.

Use the keyword `all` to configure limits that apply to each user in a cluster. If user groups are configured, the limit applies to each member of the user group, not the group as a whole.

Use the not operator (~) to exclude users or user groups from the `all` specification in the limit. This is useful if you have a large number of users but only want to exclude a few users from the limit definition.

In vertical tabular format, multiple user names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate user or user group member. Fields cannot be left blank.

Default None. If no limit is specified for PER_USER or USERS, no limit is enforced on any user or user group.

Example `PER_USER=user1 user2 ugroup1 ~user3`

PROJECTS

Syntax **PROJECTS=all** [**~**]*project_name* ..

PROJECTS
([**-**] | **all** [**~**]*project_name* ...**)**

Description A space-separated list of project names on which limits are enforced. Limits are enforced on all projects listed.

To specify a per-project limit, use the PER_PROJECT keyword. Do not configure PROJECTS and PER_PROJECT limits in the same Limit section.

In horizontal format, use only one PROJECTS line per Limit section.

Use the keyword `all` to configure limits that apply to all projects in a cluster.

Use the not operator (~) to exclude projects from the `all` specification in the limit. This is useful if you have a large number of projects but only want to exclude a few projects from the limit definition.

In vertical tabular format, multiple project names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate all projects. Fields cannot be left blank.

Default `all` (limits are enforced on all projects in the cluster)

Example `PROJECTS=projA projB`

QUEUES

Syntax **QUEUES=all** [**~**]*queue_name* ...

QUEUES
([**-**] | **all** [**~**]*queue_name* ...**)**

Description A space-separated list of queue names on which limits are enforced. Limits are enforced on all queues listed.

The list must contain valid queue names defined in `lsb.queues`.

To specify a per-queue limit, use the PER_QUEUE keyword. Do not configure QUEUES and PER_QUEUE limits in the same Limit section.

In horizontal format, use only one QUEUES line per Limit section.

Use the keyword `all` to configure limits that apply to all queues in a cluster.

Use the not operator (~) to exclude queues from the `all` specification in the limit. This is useful if you have a large number of queues but only want to exclude a few queues from the limit definition.

In vertical tabular format, multiple queue names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate all queues. Fields cannot be left blank.

Default `all` (limits are enforced on all queues in the cluster)

Example `QUEUES=normal night`

RESOURCE

Syntax **RESOURCE=[***shared_resource***,***integer***]** [**[***shared_resource***,***integer***]** ...]

RESOURCE
([**[***shared_resource***,***integer***]** [**[***shared_resource***,***integer***]** ...]**)**

Description Maximum amount of any user-defined shared resource available to consumers.

The RESOURCE keyword is a synonym for the LICENSE keyword. You can use RESOURCE to configure software licenses. You cannot specify RESOURCE and LICENSE in the same Limit section.

In horizontal format, use only one RESOURCE line per Limit section.

In vertical tabular format, resource names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses `()` or a dash (`-`) to indicate all queues. Fields cannot be left blank.

Default

None

Examples

```
RESOURCE=[stat_shared,4]
```

```
Begin Limit
RESOURCE                        PER_HOST
([stat_shared,4])               (all ~hostA)
([dyn_rsrc,1] [stat_rsrc,2])   (hostA)
End Limit
```

SLOTS

Syntax

SLOTS=*integer*

SLOTS
- | *integer*

Description

Maximum number of job slots available to resource consumers. Specify a positive integer greater than or equal 0.

With MultiCluster resource lease model, this limit applies only to local hosts being used by the local cluster. The job slot limit for hosts exported to a remote cluster is determined by the host export policy, not by this parameter. The job slot limit for borrowed hosts is determined by the host export policy of the remote cluster.

If HOSTS are configured in the Limit section, SLOTS is the number of running and suspended jobs on a host. If preemptive scheduling is used, the suspended jobs are not counted as using a job slot.

To fully use the CPU resource on multiprocessor hosts, make the number of job slots equal to or greater than the number of processors.

Use this parameter to prevent a host from being overloaded with too many jobs, and to maximize the throughput of a machine.

If only QUEUES are configured in the Limit section, SLOTS is the maximum number of job slots available to the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, SLOTS is the maximum number of job slots that the users or user groups can use on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, SLOTS is the maximum number of job slots that are available to the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, SLOTS is the maximum number of job slots that are available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit job slots per processor available to resource consumers.

In horizontal format, use only one SLOTS line per Limit section.

In vertical format, use empty parentheses `()` or a dash (`-`) to indicate the default value (no limit). Fields cannot be left blank.

Default No limit

Example `SLOTS=20`

SLOTS_PER_PROCESSOR

Syntax **SLOTS_PER_PROCESSOR=***number*

SLOTS_PER_PROCESSOR
- | *number*

Description Per processor job slot limit, based on the number of processors on each host affected by the limit.

Maximum number of job slots that each resource consumer can use per processor. This job slot limit is configured per processor so that multiprocessor hosts will automatically run more jobs.

You must also specify PER_HOST and list the hosts that the limit is to be enforced on. The Limit section is ignored if SLOTS_PER_PROCESSOR is specified:

- Without PER_HOST

OR

- With HOSTS

In vertical format, use empty parentheses `()` or a dash (`-`) to indicate the default value (no limit). Fields cannot be left blank.

To fully use the CPU resource on multiprocessor hosts, make the number of job slots equal to or greater than the number of processors.

Use this parameter to prevent a host from being overloaded with too many jobs, and to maximize the throughput of a machine.

This number can be a fraction such as 0.5, so that it can also serve as a per-CPU limit on multiprocessor machines. This number is rounded up to the nearest integer equal to or greater than the total job slot limits for a host. For example, if `SLOTS_PER_PREOCESSOR` is 0.5, on a 4-CPU multiprocessor host, users can only use up to 2 job slots at any time. On a single-processor machine, users can use 1 job slot.

If the number of CPUs in a host changes dynamically, `mbatchd` adjusts the maximum number of job slots per host accordingly. Allow the `mbatchd` up to 10 minutes to get the number of CPUs for a host. During this period the number of CPUs is 1.

If only QUEUES and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed queues for any hosts, users, or projects.

If only USERS and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor that the users or user groups can use on any hosts, queues, or projects.

If only PER_HOST is configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed hosts for any users, queues, or projects.

If only PROJECTS and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit job slots per processor available to resource consumers.

Default No limit

Example `SLOTS_PER_PROCESSOR=2`

SWP

Syntax **SWP=***integer*[**%**]

SWP
- | *integer*[**%**]

Description Maximum amount of swap space available to resource consumers. Specify a value in MB or a percentage (%) as a positive integer greater than or equal 0. If you specify a percentage, you must also specify PER_HOST and list the hosts that the limit is to be enforced on.

The Limit section is ignored if SWP is specified as a percentage:

- Without PER_HOST

OR

- With HOSTS

In horizontal format, use only one SWP line per Limit section.

In vertical format, use empty parentheses `()` or a dash (`-`) to indicate the default value (no limit). Fields cannot be left blank.

If only QUEUES are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space that the users or user groups can use on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit swap space available to resource consumers.

Default No limit

Example `SWP=60`

TMP

Syntax **TMP=***integer*[**%**]

TMP
- | *integer*[**%**]

Description Maximum amount of `tmp` space available to resource consumers. Specify a value in MB or a percentage (%) as a positive integer greater than or equal 0. If you specify a percentage, you must also specify PER_HOST and list the hosts that the limit is to be enforced on.

The Limit section is ignored if TMP is specified as a percentage:

- Without PER_HOST

OR

- With HOSTS

In horizontal format, use only one TMP line per Limit section.

In vertical format, use empty parentheses `()` or a dash (`-`) to indicate the default value (no limit). Fields cannot be left blank.

If only QUEUES are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of `tmp` space available to the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of `tmp` space that the users or user groups can use on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of `tmp` space available to the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of `tmp` space available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit `tmp` space available to resource consumers.

Default No limit

Example `TMP=20%`

USERS

Syntax **USERS=all** [**~**]*user_name* ... | **all** [**~**]*user_group* ...

USERS
([**-**] | **all** [**~**]*user_name* ... | **all** [**~**]*user_group* ... **)**

Description A space-separated list of user names or user groups on which limits are enforced. Limits are enforced on all users or groups listed. Limits apply to a group as a whole.

If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups.

To specify a per-user limit, use the PER_USER keyword. Do not configure USERS and PER_USER limits in the same Limit section.

In horizontal format, use only one USERS line per Limit section.

Use the keyword `all` to configure limits that apply to all users or user groups in a cluster.

Use the not operator (~) to exclude users or user groups from the `all` specification in the limit. This is useful if you have a large number of users but only want to exclude a few users or groups from the limit definition.

In vertical format, multiple user names must be enclosed in parentheses.

In vertical format, use empty parentheses `()` or a dash `(-)` to indicate all users or groups. Fields cannot be left blank.

Default `all` (limits are enforced on all users in the cluster)

Example `USERS=user1 user2`

HostExport Section

Defines an export policy for a host or a group of related hosts. Defines how much of each host's resources are exported, and how the resources are distributed among the consumers.

Each export policy is defined in a separate HostExport section, so it is normal to have multiple HostExport sections in `lsb.resources`.

Example HostExport section

```
Begin HostExport
PER_HOST= hostA hostB
SLOTS= 4
DISTRIBUTION= [cluster1, 1] [cluster2, 3]
MEM= 100
SWAP= 100
End HostExport
```

HostExport section structure

Use empty parentheses ( ) or a dash (-) to specify the default value for an entry. Fields cannot be left blank.

PER_HOST

Syntax **PER_HOST=***host_name*...

Description Required when exporting special hosts.

Determines which hosts to export. Specify one or more LSF hosts by name. Separate names by space.

RES_SELECT

Syntax **RES_SELECT=***res_req*

Description Required when exporting workstations.

Determines which hosts to export. Specify the selection part of the resource requirement string (without quotes or parentheses), and LSF will automatically select hosts that meet the specified criteria. For this parameter, if you do not specify the required host type, the default is "`type==any`".

The criteria is only evaluated once, when a host is exported.

NHOSTS

Syntax **NHOSTS=***integer*

Description Required when exporting workstations.

Maximum number of hosts to export. If there are not this many hosts meeting the selection criteria, LSF exports as many as it can.

DISTRIBUTION

Syntax **DISTRIBUTION=([***cluster_name***,** *number_shares***]**...**)**

Description Required. Specifies how the exported resources are distributed among consumer clusters.

The syntax for the distribution list is a series of share assignments. The syntax of each share assignment is the cluster name, a comma, and the number of shares, all enclosed in square brackets, as shown. Use a space to separate multiple share assignments. Enclose the full distribution list in a set of round brackets.

- *cluster_name*

  Specify the name of a remote cluster that will be allowed to use the exported resources.

  If you specify a local cluster, the assignment is ignored.

- *number_shares*

  Specify a positive integer representing the number of shares of exported resources assigned to the cluster.

  The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters, or to the total number. The total number of shares is just the sum of all the shares assigned in each share assignment.

MEM

Syntax **`MEM=`***megabytes*

Description Used when exporting special hosts. Specify the amount of memory to export on each host, in MB.

Default - (provider and consumer clusters compete for available memory)

SLOTS

Syntax **`SLOTS=`***integer*

Description Required when exporting special hosts. Specify the number of job slots to export on each host.

To avoid overloading a partially exported host, you can reduce the number of job slots in the configuration of the local cluster.

SWAP

Syntax **`SWAP=`***megabytes*

Description Used when exporting special hosts. Specify the amount of swap space to export on each host, in MB.

Default - (provider and consumer clusters compete for available swap space)

TYPE

Syntax **`TYPE=shared`**

Description Changes the lease type from exclusive to shared.

If you export special hosts with a shared lease (using PER_HOST), you cannot specify multiple consumer clusters in the distribution policy.

Default Undefined (the lease type is exclusive; exported resources are never available to the provider cluster)

SharedResourceExport Section

Optional. Requires HostExport section. Defines an export policy for a shared resource. Defines how much of the shared resource is exported, and the distribution among the consumers.

The shared resource must be available on hosts defined in the HostExport sections.

Example SharedResourceExport section

```
Begin SharedResourceExport
NAME= AppLicense
NINSTANCES= 10
DISTRIBUTION= ([C1, 30] [C2, 70])
End SharedResourceExport
```

SharedResourceExport section structure

All parameters are required.

NAME

Syntax **NAME=***shared_resource_name*

Description Shared resource to export. This resource must be available on the hosts that are exported to the specified clusters; you cannot export resources without hosts.

NINSTANCES

Syntax **NINSTANCES=***integer*

Description Maximum quantity of shared resource to export. If the total number available is less than the requested amount, LSF exports all that are available.

DISTRIBUTION

Syntax **DISTRIBUTION=([***cluster_name***,** *number_shares***]**...**)**

Description Specifies how the exported resources are distributed among consumer clusters.

The syntax for the distribution list is a series of share assignments. The syntax of each share assignment is the cluster name, a comma, and the number of shares, all enclosed in square brackets, as shown. Use a space to separate multiple share assignments. Enclose the full distribution list in a set of round brackets.

cluster_name

Specify the name of a cluster allowed to use the exported resources.

number_shares

Specify a positive integer representing the number of shares of exported resources assigned to the cluster.

The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters, or to the total number. The total number of shares is the sum of all the shares assigned in each share assignment.

ResourceReservation Section

By default, only LSF administrators or root can add or delete advance reservations.

The ResourceReservation section defines an advance reservation policy. It specifies:

- Users or user groups that can create reservations
- Hosts that can be used for the reservation
- Time window when reservations can be created

Each advance reservation policy is defined in a separate ResourceReservation section, so it is normal to have multiple ResourceReservation sections in `lsb.resources`.

Example ResourceReservation section

Only `user1` and `user2` can make advance reservations on `hostA` and `hostB`. The reservation time window is between 8:00 a.m. and 6:00 p.m. every day:

```
Begin ResourceReservation
NAME        = dayPolicy
USERS       = user1 user2     # optional
HOSTS       = hostA hostB     # optional
TIME_WINDOW = 8:00-18:00      # weekly recurring reservation
End ResourceReservation
```

`user1` can add the following reservation for user `user2` to use on `hostA` every Friday between 9:00 a.m. and 11:00 a.m.:

```
% user1@hostB> brsvadd -m "hostA" -n 1 -u "user2" -t "5:9:0-
5:11:0"
Reservation "user2#2" is created
```

Users can only delete reservations they created themselves. In the example, only user `user1` can delete the reservation; `user2` cannot. Administrators can delete any reservations created by users.

HOSTS

Syntax **HOSTS=**[~]*host_name* | [~]*host_group* | **all** | **allremote** | **all@cluster_name** ...

Description A space-separated list of hosts, host groups defined in `lsb.hosts` on which administrators or users specified in the USERS parameter can create advance reservations.

The hosts can be local to the cluster or hosts leased from remote clusters.

If a group contains a subgroup, the reservation configuration applies to each member in the subgroup recursively.

Use the keyword `all` to configure reservation policies that apply to all local hosts in a cluster not explicitly excluded. This is useful if you have a large cluster but you want to use the not operator (~) to exclude a few hosts from the list of hosts where reservations can be created.

Use the keyword `allremote` to specify all hosts borrowed from all remote clusters.

You cannot specify host groups or host partitions that contain the `allremote` keyword.

Use `all@cluster_name` to specify the group of all hosts borrowed from one remote cluster. You cannot specify a host group or partition that includes remote resources.

With MultiCluster resource leasing model, the not operator (~) can be used to exclude local hosts or host groups. You cannot use the not operator (~) with remote hosts.

Examples

- `HOSTS=hgroup1 ~hostA hostB hostC`

  Advance reservations can be created on `hostB`, `hostC`, and all hosts in `hgroup1` except for `hostA`.
- `HOSTS=all ~group2 ~hostA`

  Advance reservations can be created on all hosts in the cluster, except for `hostA` and the hosts in `group2`.

Default `all allremote` (users can create reservations on all server hosts in the local cluster, and all leased hosts in a remote cluster).

NAME

Syntax **NAME=***text*

Description Required. Name of the ResourceReservation section

Specify any ASCII string 40 characters or less. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

Example `NAME=reservation1`

Default None. You must provide a name for the ResourceReservation section.

TIME_WINDOW

Syntax **TIME_WINDOW=***time_window* ...

Description Optional. Time window for users to create advance reservations. The time for reservations that users create must fall within this time window.

Use the same format for *time_window* as the recurring reservation option (`-t`) of `brsvadd`:

[*day*:]*hour*[:*minute*]

with the following ranges:

- *day of the week*: 0-6
- *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*

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.

You can specify multiple time windows, but they cannot overlap. For example:

`TIME_WINDOW=8:00-14:00 18:00-22:00`

is correct, but

`TIME_WINDOW=8:00-14:00 11:00-15:00`

is not valid.

Example `TIME_WINDOW=8:00-14:00`

Users can create advance reservations with begin time (`brsvadd -b`), end time (`brsvadd -e`), or time window (`brsvadd -t`) on any day between 8:00 a.m. and 2:00 p.m.

Default Undefined (any time)

USERS

Syntax **USERS=**[**~**]*user_name* | [**~**]*user_group* ... | **all**

Description A space-separated list of user names or user groups who are allowed to create advance reservations. Administrators, root, and all users or groups listed can create reservations.

If a group contains a subgroup, the reservation policy applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups.

Use the keyword `all` to configure reservation policies that apply to all users or user groups in a cluster. This is useful if you have a large number of users but you want to exclude a few users or groups from the reservation policy.

Use the not operator (~) to exclude users or user groups from the list of users who can create reservations.

The not operator does not exclude LSF administrators from the policy.

Example `USERS=user1 user2`

Default `all` (all users in the cluster can create reservations)

ReservationUsage Section

To enable greater flexibility for reserving numeric resources are reserved by jobs, configure the ReservationUsage section in `lsb.resources` to reserve resources like license tokes per resource as PER_JOB, PER_SLOT, or PER_HOST. For example:

Example ReservationUsage section

```
Begin ReservationUsage
RESOURCE             METHOD
licenseX             PER_JOB
licenseY             PER_HOST
licenseZ             PER_SLOT
End ReservationUsage
```

RESOURCE

The name of the resource to be reserved. Only user-defined numeric resources can be reserved. Builtin resources like mem, cpu, swp, etc. cannot be configured in the ReservationUsage section.

METHOD

The resource reservation method. One of:

- PER_JOB
- PER_HOST
- PER_SLOT

The cluster-wide RESOURCE_RESERVE_PER_SLOT parameter in `lsb.params` is obsolete. Configuration in `lsb.resources` overrides RESOURCE_RESERVE_PER_SLOT if it also exists for the same resource.

RESOURCE_RESERVE_PER_SLOT parameter still controls resources not configured in `lsb.resources`. Resources not reserved in `lsb.resources` are reserved per job.

PER_HOST reservation means that for the parallel job, LSF reserves one instance of a for each host. For example, some application licenses are charged only once no matter how many applications are running provided those applications are running on the same host under the same user.

Assumptions and limitations

- Per-resource configuration defines resource usage for individual resources, but it does not change any existing resource limit behavior (PER_JOB, PER_SLOT).
- In a MultiCluster environment, you should configure resource usage in the scheduling cluster (submission cluster in lease model or receiving cluster in job forward model).

Automatic Time-based Configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.resources` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the `badmin reconfig` command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

```
# limit usage of hosts in 'license1' group and time
# based configuration
# - 10 jobs can run from normal queue
# - any number can run from short queue between 18:30
#   and 19:30
#   all other hours you are limited to 100 slots in the
#   short queue
# - each other queue can run 30 jobs

Begin Limit
PER_QUEUE               HOSTS       SLOTS     # Example
normal                  license1    10
# if time(18:30-19:30)
short                   license1    -
#else
short                   license1    100
#endif
(all ~normal ~short)    license1    30
End Limit
```

SEE ALSO

badmin(8), blimits(1), brsvadd(1), bsub(1), lsb.hosts(5), lsb.queues(5), lsb.users(5)

lsb.serviceclasses

The `lsb.serviceclasses` file defines the service-level agreements (SLAs) in an LSF cluster as *service classes*, which define the properties of the SLA.

This file is optional.

You can configure as many service class sections as you need.

Use `bsla` to display the properties of service classes configured in `lsb.serviceclasses` and dynamic information about the state of each configured service class.

By default, `lsb.serviceclasses` is installed in `LSB_CONFDIR/`*`cluster_name`*`/configdir`.

Changing lsb.serviceclasses configuration

After making any changes to `lsb.serviceclasses`, run `badmin reconfig` to reconfigure `mbatchd`.

Contents

- "lsb.serviceclasses structure" on page 458

lsb.serviceclasses structure

Each service class definition begins with the line `Begin ServiceClass` and ends with the line `End ServiceClass`.

Syntax

```
Begin ServiceClass
NAME           = string
PRIORITY       = integer
GOALS          = [throughput | velocity | deadline] [\
                 [throughput | velocity | deadline] ...]
CONTROL_ACTION = VIOLATION_PERIOD[minutes] CMD [action]
USER_GROUP     = all | [user_name] [user_group] ...
DESCRIPTION    = text
```

You must specify:

- Service class name
- Goals
- Priority

All other parameters are optional.

Example

```
Begin ServiceClass
NAME=Uclulet
PRIORITY=20
GOALS=[DEADLINE timeWindow (8:30-16:00)]
DESCRIPTION="working hours"
End ServiceClass
```

CONTROL_ACTION

Syntax

CONTROL_ACTION=VIOLATION_PERIOD[*minutes***] CMD [***action***]**

Description

Optional. Configures a control action to be run if the SLA goal is delayed for a specified number of minutes.

If the SLA goal is delayed for longer than VIOLATION_PERIOD, the action specified by CMD is invoked. The violation period is reset and if the SLA is still active when the violation period expires again, the action runs again. If the SLA has multiple active goals that are in violation, the action is run for each of them.

Example

```
CONTROL_ACTION=VIOLATION_PERIOD[10] CMD [echo `date`: SLA is
in violation >> ! /tmp/sla_violation.log]
```

Default

None

DESCRIPTION

Syntax

DESCRIPTION=*text*

Description

Optional. Description of the service class. Use `bsla` to display the description text.

This description should clearly describe the features of the service class to help users select the proper service class for their jobs.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 512 characters.

Default None

GOALS

Syntax **GOALS=[***throughput* | *velocity* | *deadline***]** [****
[*throughput* | *velocity* | *deadline***]** ...]

Description Required. Defines the service-level goals for the service class. A service class can have more than one goal, each active at different times of the day and days of the week. Outside of the time window, the SLA is inactive and jobs are scheduled as if no service class is defined. LSF does not enforce any service-level goal for an inactive SLA.

The time windows of multiple service-level goals can overlap. In this case, the largest number of jobs is run.

An active SLA can have a status of `On time` if it is meeting the goal, and a status `Delayed`, if it is missing its goals.

A service-level goal defines:

- *throughput*—expressed as *finished* jobs per hour and an optional time window when the goal is active. *throughput* has the form:

  GOALS=[THROUGHPUT *num_jobs* **timeWindow** [**(***time_window***)**]**]**

  If no time window is configured, THROUGHPUT can be the only goal in the service class. The service class is always active, and `bsla` displays `ACTIVE WINDOW: Always Open`.
- *velocity*—expressed as *concurrently* running jobs and an optional time window when the goal is active. *velocity* has the form:

  GOALS=[VELOCITY *num_jobs* **timeWindow** [**(***time_window***)**]**]**

  If no time window is configured, VELOCITY can be the only goal in the service class. The service class is always active, and `bsla` displays `ACTIVE WINDOW: Always Open`.
- *deadline*—indicates that all jobs in the service class should complete by the end of the specified time window. The time window is required for a deadline goal. *deadline* has the form:

  GOALS=[DEADLINE timeWindow (*time_window***)]**

Time window format The time window of an SLA goal has the standard form:

[*day***:**]*hour*[**:***minute*]

with the following ranges:

- *day of the week*: 0-6
- *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*

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.

You can specify multiple time windows, but they cannot overlap. For example:

```
timeWindow(8:00-14:00 18:00-22:00)
```

is correct, but

```
timeWindow(8:00-14:00 11:00-15:00)
```

is not valid.

To configure a time window that is always open, use the timeWindow keyword with empty parentheses.

Examples

```
GOALS=[THROUGHPUT 2 timeWindow ()]
GOALS=[THROUGHPUT 10 timeWindow (8:30-16:30)]
GOALS=[VELOCITY 5 timeWindow ()]
GOALS=[DEADLINE timeWindow (16:30-8:30)]\
        [VELOCITY 10 timeWindow (8:30-16:30)]
```

NAME

Syntax

NAME=*string*

Description

Required. A unique name that identifies the service class.

Specify any ASCII string 60 characters or less. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

IMPORTANT

The name you use cannot be the same as an existing host partition or user group name.

Example

`NAME=Tofino`

Default

None. You must provide a unique name for the service class.

PRIORITY

Syntax

PRIORITY=*integer*

Description

Required. 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.

LSF schedules jobs from one service class at a time, starting with the highest-priority service class. If multiple service classes have the same priority, LSF runs all the jobs from these service classes in first-come, first-served order.

Service class priority in LSF is completely independent of the UNIX scheduler's priority system for time-sharing processes. In LSF, the NICE parameter is used to set the UNIX time-sharing priority for batch jobs.

Default

1 (lowest possible priority)

USER_GROUP

Syntax **USER_GROUP=all** | [*user_name*] [*user_group*] ...

Description Optional. A space-separated list of user names or user groups who can submit jobs to the service class. Administrators, root, and all users or groups listed can use the service class.

Use the reserved word `all` to specify all LSF users. LSF cluster administrators are automatically included in the list of users, so LSF cluster administrators can submit jobs to any service class, or switch any user's jobs into this service class, even if they are not listed.

If user groups are specified in `lsb.users`, each user in the group can submit jobs to this service class. If a group contains a subgroup, the service class policy applies to each member in the subgroup recursively. If the group can define fairshare among its members, the SLA defined by the service class enforces the fairshare policy among the users of the SLA.

User names must be valid login names. User group names can be LSF user groups (in `lsb.users`) or UNIX and Windows user groups.

Example `USER_GROUP=user1 user2 ugroup1`

Default `all` (all users in the cluster can submit jobs to the service class)

Examples

- The service class `Uclulet` defines one deadline goal that is active during working hours between 8:30 AM and 4:00 PM. All jobs in the service class should complete by the end of the specified time window. Outside of this time window, the SLA is inactive and jobs are scheduled without any goal being enforced:

```
Begin ServiceClass
NAME=Uclulet
PRIORITY=20
GOALS=[DEADLINE timeWindow (8:30-16:00)]
DESCRIPTION="working hours"
End ServiceClass
```

- The service class `Nanaimo` defines a deadline goal that is active during the weekends and at nights.

```
Begin ServiceClass
NAME=Nanaimo
PRIORITY=20
GOALS=[DEADLINE timeWindow (5:18:00-1:8:30 20:00-8:30)]
DESCRIPTION="weekend nighttime regression tests"
End ServiceClass
```

- The service class `Inuvik` defines a throughput goal of 6 jobs per hour that is always active:

```
Begin ServiceClass
NAME=Inuvik
PRIORITY=20
GOALS=[THROUGHPUT 6 timeWindow ()]
DESCRIPTION="constant throughput"
End ServiceClass
```

- The service class `Tofino` defines two velocity goals in a 24 hour period. The first goal is to have a maximum of 10 concurrently running jobs during business hours (9:00 a.m. to 5:00 p.m). The second goal is a maximum of 30 concurrently running jobs during off-hours (5:30 p.m. to 8:30 a.m.)

```
Begin ServiceClass
NAME=Tofino
PRIORITY=20
GOALS=[VELOCITY 10 timeWindow (9:00-17:00)] \
      [VELOCITY 30 timeWindow (17:30-8:30)]
DESCRIPTION="day and night velocity"
End ServiceClass
```

- The service class `Kyuquot` defines a velocity goal that is active during working hours (9:00 a.m. to 5:30 p.m.) and a deadline goal that is active during off-hours (5:30 p.m. to 9:00 a.m.) Only users `user1` and `user2` can submit jobs to this service class.

```
Begin ServiceClass
NAME=Kyuquot
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 service class `Tevere` defines a combination similar to `Kyuquot`, but with a deadline goal that takes effect overnight and on weekends. During the working hours in weekdays the velocity goal favors a mix of short and medium jobs.

```
Begin ServiceClass
NAME=Tevere
PRIORITY=20
GOALS=[VELOCITY 100 timeWindow (9:00-17:00)] \
      [DEADLINE timeWindow (17:30-8:30 5:17:30-1:8:30)]
DESCRIPTION="nine to five"
End ServiceClass
```

SEE ALSO

`bacct`(1), `bhist`(1), `bjobs`(1), `bkill`(1), `bmod`(1), `bsla`(1), `bsub`(1), `lsb.users`(5)

SEE ALSO

lsb.users

The `lsb.users` file is used to configure user groups, hierarchical fairshare for users and user groups, and job slot limits for users and user groups. It is also used to configure account mappings in a MultiCluster environment.

This file is optional.

The `lsb.users` file is stored in the directory `LSB_CONFDIR/`*`cluster_name`*`/configdir`, where LSB_CONFDIR is defined in `lsf.conf`.

Changing lsb.users configuration

After making any changes to `lsb.users`, run `badmin reconfig` to reconfigure `mbatchd`.

Contents

- "UserGroup Section" on page 466
- "User Section" on page 468
- "UserMap Section" on page 470
- "Automatic Time-based Configuration" on page 472

UserGroup Section

Optional. Defines user groups.

The name of the user group can be used in other user group and queue definitions, as well as on the command line. Specifying the name of a user group has exactly the same effect as listing the names of all users in the group.

The total number of user groups cannot be more than MAX_GROUPS in `lsbatch.h`.

Structure

The first line consists of two mandatory keywords, `GROUP_NAME` and `GROUP_MEMBER`. The `USER_SHARES` keyword is optional. Subsequent lines name a group and list its membership and optionally its share assignments.

Each line must contain one entry for each keyword. Use empty parentheses `()` or a dash `-` to specify the default value for an entry.

Example of a UserGroup Section

```
Begin UserGroup
GROUP_NAME    GROUP_MEMBER
groupA        (user1 user2 user3 user4)
groupB        (groupA user5)
groupC        (!)
End UserGroup
```

```
Begin UserGroup
GROUP_NAME    GROUP_MEMBER                USER_SHARES
groupB        (user1 user2)               ()
groupC        (user3 user4)               ([User3,3] [User4,4])
groupA        (GroupB GroupC user5)       ([User5,1] [default,10])
End UserGroup
```

GROUP_NAME

An alphanumeric string representing the user group name. You cannot use the reserved name `all` or a `/` in a group name`,` and group names must not conflict with user names.

GROUP_MEMBER

A list of user names or user group names that belong to the group, enclosed in parentheses and separated by spaces. Group names must not conflict with user names.

User and user group names can appear on multiple lines, because users can belong to multiple groups.

User groups may be defined recursively but must not create a loop.

Syntax

(*user_name* | *user_group* ...**)** | **(all)** | **(!)**

Specify the following, all enclosed in parentheses:

- *user_name* | *user_group*

  User and user group names, separated by spaces. User names must be valid login names.

User group names can be LSF user groups defined previously in this section, or UNIX and Windows user groups.

- **`all`**

  The reserved name `all` specifies all users in the cluster.

- **`!`**

  The exclamation mark `!` specifies that the group membership should be retrieved via `egroup`.

USER_SHARES

Optional. Enables hierarchical fairshare and defines a share tree for users and user groups.

By default, when resources are assigned collectively to a group, the group members compete for the resources according to FCFS scheduling. You can use hierarchical fairshare to further divide the shares among the group members.

Syntax

([*user***,** *number_shares***])**

Specify the arguments as follows:

- Enclose the list in parentheses, even if you do not specify any user share assignments.
- Enclose each user share assignment in square brackets, as shown.
- Separate the list of share assignments with a space.
- *user*

  Specify users or user groups. You can assign the shares to:

  - A single user (specify *user_name*)
  - Users in a group, individually (specify *group_name***@**) or collectively (specify *group_name*)
  - Users not included in any other share assignment, individually (specify the keyword `default`) or collectively (specify the keyword `others`)

  By default, when resources are assigned collectively to a group, the group members compete for the resources on a first-come, first-served (FCFS) basis. You can use hierarchical fairshare to further divide the shares among the group members.

  When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*

  Specify a positive integer representing the number of shares of the cluster resources assigned to the user.

  The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

User Section

Optional. If this section is not defined, all users and user groups can run an unlimited number of jobs in the cluster.

This section defines the maximum number of jobs a user or user group can run concurrently in the cluster. This is to avoid situations in which a user occupies all or most of the system resources while other users' jobs are waiting.

Structure

Three fields are mandatory: USER_NAME, MAX_JOBS, JL/P.

MAX_PEND_JOBS is optional.

You must specify a dash (-) to indicate the default value (unlimited) if a user or user group is specified. Fields cannot be left blank.

Example of a User Section

```
Begin User
USER_NAME    MAX_JOBS    JL/P    MAX_PEND_JOBS
user1        10           -         1000
user2         4           -            -
user3         -           -            -
groupA       10           1       100000
groupA@       -           1          100
groupC        -           -          500
default       6           1           10
End User
```

USER_NAME

User or user group for which job slot limits are defined.

Use the reserved user name default to specify a job slot limit that applies to each user and user group not explicitly named. Since the limit specified with the keyword default applies to user groups also, make sure you select a limit that is high enough, or explicitly define limits for user groups.

User group names can be the LSF user groups defined previously, and/or UNIX and Windows user groups.

Job slot limits apply to a group as a whole. Append the at sign (@) to a group name to make the job slot limits apply individually to each user in the group. If a group contains a subgroup, the job slot limit also applies to each member in the subgroup recursively.

If the group contains the keyword all in the user list, the at sign (@) has no effect. To specify job slot limits for each user in a user group containing all, use the keyword default.

MAX_JOBS

Per-user or per-group job slot limit for the cluster. Total number of job slots that each user or user group can use in the cluster.

JL/P

Per processor job slot limit per user or user group.

Total number of job slots that each user or user group can use per processor. This job slot limit is configured per processor so that multiprocessor hosts will automatically run more jobs.

This number can be a fraction such as 0.5, so that it can also serve as a per-host limit. This number is rounded up to the nearest integer equal to or greater than the total job slot limits for a host. For example, if `JL/P` is 0.5, on a 4-CPU multiprocessor host, the user can only use up to 2 job slots at any time. On a uniprocessor machine, the user can use 1 job slot.

MAX_PEND_JOBS

Per-user or per-group pending job limit. This is the total number of pending job slots that each user or user group can have in the system. If a user is a member of multiple user groups, the user's pending jobs are counted towards the pending job limits of all groups from which the user has membership.

UserMap Section

Optional. Used only in a MultiCluster environment.

Defines system-level account mapping for users and user groups.

To support the execution of batch jobs across non-uniform user name spaces between clusters, LSF allows user account mapping. For a job submitted by one user account in one cluster to run under a different user account in a remote cluster, both the local and remote clusters must have the account mapping properly configured.

Structure

All three fields `LOCAL`, `REMOTE` and `DIRECTION` are required.

Example of a UserMap Section

On cluster1

```
Begin UserMap
LOCAL    REMOTE                              DIRECTION
user1    user2@cluster2                      export
user3    (user4@cluster2 user6@cluster2)     export
End UserMap
```

On cluster2

```
Begin UserMap
LOCAL             REMOTE                     DIRECTION
user2             user1@cluster1             import
(user6 user8)     user3@cluster1             import
End UserMap
```

`Cluster1` configures `user1` to run jobs as `user2` and `user3` to run jobs as `user4` or `user6`.

`Cluster2` configures `user1` to run jobs as `user2` and `user3` to run jobs as `user6` or `user8`.

Only mappings configured in both clusters will work. The common account mappings are for `user1` to run jobs as `user2` and for `user3` to run jobs as `user6`. Therefore, these mappings will work, but the mappings of `user3` to `user4` and `user8` are only half-done and so will not work.

LOCAL

A list of users or user groups in the local cluster.

Multiple user names and user group names must be separated by a space, and the entire list enclosed in parentheses `()`.

REMOTE

A list of remote users or user groups in the form: `user_name@cluster_name`
`user_group_name@cluster_name`

Multiple user names and user group names must be separated by a space, and the entire list enclosed in parentheses `()`.

DIRECTION

Configures the direction of account mapping:

- The `export` keyword configures local users/groups to run jobs as remote users/groups.
- The `import` keyword configures remote users/groups to run jobs as local users/groups.

Both directions must be configured for a mapping to work. The mapping must be configured in both the local and remote clusters.

Automatic Time-based Configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.users` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the `badmin reconfig` command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example From 12 - 1 p.m. daily, user `smith` has 10 job slots, but during other hours, user has only 5 job slots.

```
Begin User
USER_NAMEMAX_JOBSJL/P
#if time (12-13)
smith10-
#else
smith5-
default1-
#endif
End User
```

SEE ALSO

`lsf.cluster`(5), `lsf.conf`(5), `lsb.params`(5), `lsb.hosts`(5), `lsb.queues`(5), `bhosts`(1), `bmgroup`(1), `bhpart`(1), `busers`(1), `bugroup`(1), `bqueues`(1), `bsub`(1), `bchkpnt`(1), `lsid`(1), `nice`(1), `getgrnam`(3), `mbatchd`(8), `badmin`(8)

lsf.acct

The `lsf.acct` file is the LSF task log file.

The LSF Remote Execution Server, RES (see `res`(8)), generates a record for each task completion or failure. If the RES task logging is turned on (see `lsadmin`(8)), it appends the record to the task log file `lsf.acct.`<*`host_name`*>.

Contents

- "lsf.acct Structure" on page 476

lsf.acct Structure

The task log file is an ASCII file with one task record per line. The fields of each record are separated by blanks. The location of the file is determined by the LSF_RES_ACCTDIR variable defined in `lsf.conf`. If this variable is not defined, or the RES cannot access the log directory, the log file is created in `/tmp` instead.

Fields

The fields in a task record are ordered in the following sequence:

`pid (%d)`

Process ID for the remote task

`userName (%s)`

User name of the submitter

`exitStatus (%d)`

Task exit status

`dispTime (%ld)`

Dispatch time – time at which the task was dispatched for execution

`termTime (%ld)`

Completion time – time when task is completed/failed

`fromHost (%s)`

Submission host name

`execHost (%s)`

Execution host name

`cwd (%s)`

Current working directory

`cmdln (%s)`

Command line of the task

`lsfRusage`

The following fields contain resource usage information for the job (see `getrusage`(2)). If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

`ru_utime (%f)`

User time used

`ru_stime (%f)`

System time used

`ru_maxrss (%f)`

Maximum shared text size

`ru_ixrss (%f)`

Integral of the shared text size over time (in KB seconds)

`ru_ismrss (%f)`

Integral of the shared memory size over time (valid only on Ultrix)

`ru_idrss (%f)`

Integral of the unshared data size over time

`ru_isrss (%f)`

Integral of the unshared stack size over time

`ru_minflt (%f)`

Number of page reclaims

`ru_majflt (%f)`

Number of page faults

`ru_nswap (%f)`

Number of times the process was swapped out

`ru_inblock (%f)`

Number of block input operations

`ru_oublock (%f)`

Number of block output operations

`ru_ioch (%f)`

Number of characters read and written (valid only on HP-UX)

`ru_msgsnd (%f)`

Number of System V IPC messages sent

`ru_msgrcv (%f)`

Number of messages received

`ru_nsignals (%f)`

Number of signals received

`ru_nvcsw (%f)`

Number of voluntary context switches

`ru_nivcsw (%f)`

Number of involuntary context switches

`ru_exutime (%f)`

Exact user time used (valid only on ConvexOS)

SEE ALSO

Related Topics: lsadmin(8), res(8), lsf.conf(5), getrusage(2)

Files: $LSF_RES_ACCTDIR/lsf.acct.*host_name*

lsf.cluster

Contents

- “About lsf.cluster” on page 480
- “Parameters Section” on page 481
- “ClusterAdmins Section” on page 489
- “Host Section” on page 491
- “ResourceMap Section” on page 495
- “RemoteClusters Section” on page 497

About lsf.cluster

This is the cluster configuration file. There is one for each cluster, called `lsf.cluster.`*`cluster_name`*. The *cluster_name* suffix is the name of the cluster defined in the Cluster section of `lsf.shared`. All LSF hosts are listed in this file, along with the list of LSF administrators and the installed LSF features.

The `lsf.cluster.`*`cluster_name`* file contains two types of configuration information:

- Cluster definition information—affects all LSF applications. Defines cluster administrators, hosts that make up the cluster, attributes of each individual host such as host type or host model, and resources using the names defined in `lsf.shared`.
- LIM policy information—affects applications that rely on LIM job placement policy. Defines load sharing and job placement policies provided by LIM.

Changing lsf.cluster configuration

After making any changes to `lsf.cluster.`*`cluster_name`*, run the following commands:

- `lsadmin reconfig` to reconfigure LIM
- `badmin mbdrestart` to restart `mbatchd`

Location

This file is typically installed in the directory defined by LSF_ENVDIR.

Structure

The `lsf.cluster.`*`cluster_name`* file contains the following configuration sections:

- "Parameters Section"
- "ClusterAdmins Section"
- "Host Section"
- "ResourceMap Section"
- "RemoteClusters Section"

Parameters Section

(Optional) This section contains miscellaneous parameters for the LIM.

ADJUST_DURATION

Syntax **ADJUST_DURATION=***integer*

Description Integer reflecting a multiple of EXINTERVAL that controls the time period during which load adjustment is in effect

The `lsplace(1)` and `lsloadadj(1)` commands artificially raise the load on a selected host. This increase in load decays linearly to 0 over time.

Default 3

ELIM_POLL_INTERVAL

Syntax **ELIM_POLL_INTERVAL=***time_in_seconds*

Description Time interval, in seconds, in which the LIM daemon samples load information

This parameter only needs to be set if an ELIM is being used to report information more frequently than every 5 seconds.

Default 5 seconds

ELIMARGS

Syntax **ELIMARGS=***cmd_line_args*

Description Specifies any necessary command-line arguments for the external LIM on startup

This parameter is ignored if no external load indices are configured.

Default None

EXINTERVAL

Syntax **EXINTERVAL=***time_in_seconds*

Description Time interval, in seconds, at which the LIM daemons exchange load information

On extremely busy hosts or networks, or in clusters with a large number of hosts, load may interfere with the periodic communication between LIM daemons. Setting EXINTERVAL to a longer interval can reduce network load and slightly improve reliability, at the cost of slower reaction to dynamic load changes.

Note that if you define the time interval as less than 5 seconds, LSF automatically resets it to 5 seconds.

Default 15 seconds

FLOAT_CLIENTS

Syntax **FLOAT_CLIENTS=***number_of_floating_client_licenses*

Description Sets the size of your license pool in the cluster

When the master LIM starts, up to *number_of_floating_client_licenses* will be checked out for use as floating client licenses. If fewer licenses are available than specified by *number_of_floating_client_licenses*, only the available licenses will be checked out and used.

If FLOAT_CLIENTS is not specified in `lsf.cluster.`*`cluster_name`* or there is an error in either `license.dat` or in `lsf.cluster.`*`cluster_name`*, the floating LSF client license feature is disabled.

WARNING **When the LSF floating client feature is enabled, any host will be able to submit jobs to the cluster. You can limit which hosts can be LSF floating clients with the parameter FLOAT_CLIENTS_ADDR_RANGE in lsf.cluster.cluster_name.**

LSF Floating Client Although LSF Floating Client requires a license, LSF_Float_Client does not need to be added to the PRODUCTS line. LSF_Float_Client also cannot be added as a resource for specific hosts already defined in lsf.cluster.cluster_name. Should these lines be present, they are ignored by LSF.

Default Undefined

FLOAT_CLIENTS_ADDR_RANGE

Syntax **`FLOAT_CLIENTS_ADDR_RANGE=`***IP_address* ...

Description Optional. IP address or range of addresses, in dotted quad notation (`nnn.nnn.nnn.nnn`), of domains from which floating client hosts can submit requests. Multiple ranges can be defined, separated by spaces.

If the value of this parameter is undefined, there is no security and any hosts can be LSF floating clients.

If a value is defined, security is enabled. If there is an error in the configuration of this variable, by default, no hosts will be allowed to be LSF floating clients.

When this parameter is defined, client hosts that do not belong to the domain will be denied access.

If a requesting host belongs to an IP address that falls in the specified range, the host will be accepted to become a floating client.

IP addresses are separated by spaces, and considered "OR" alternatives.

The asterisk (*) character indicates any value is allowed.

The dash (-) character indicates an explicit range of values. For example 1-4 indicates 1,2,3,4 are allowed.

Open ranges such as *-30, or 10-*, are allowed.

If a range is specified with less fields than an IP address such as 10.161, it is considered as 10.161.*.*.

Address ranges are validated at configuration time so they must conform to the required format. If any address range is not in the correct format, no hosts will be accepted as LSF floating clients and an error message will be logged in the LIM log.

This parameter is limited to 255 characters.

Notes After you configure FLOAT_CLIENTS_ADDR_RANGE, check the `lim.log.`*`host_name`* file to make sure this parameter is correctly set. If this parameter is not set or is wrong, this will be indicated in the log file.

Examples

- `FLOAT_CLIENTS_ADDR_RANGE=100`
  All client hosts with a domain address starting with 100 will be allowed access.
- `FLOAT_CLIENTS_ADDR_RANGE=100-110.34.1-10.4-56`
  All client hosts belonging to a domain with an address having the first number between 100 and 110, then 34, then a number between 1 and 10, then, a number between 4 and 56 will be allowed access.
  Example: 100.34.9.45, 100.34.1.4, 102.34.3.20, etc.
- `FLOAT_CLIENTS_ADDR_RANGE=100.172.1.13 100.*.30-54 124.24-*.1.*-34`
  All client hosts belonging to a domain with the address 100.172.1.13 will be allowed access. All client hosts belonging to domains starting with 100, then any number, then a range of 30 to 54 will be allowed access. All client hosts belonging to domains starting with 124, then from 24 onward, then 1, then from 0 to 34 will be allowed access.
- `FLOAT_CLIENTS_ADDR_RANGE=12.23.45.*`
  All client hosts belonging to domains starting with 12.23.45 are allowed.
- `FLOAT_CLIENTS_ADDR_RANGE=100.*43`
  The `*` character can only be used to indicate any value. In this example, an error will be inserted in the LIM log and no hosts will be accepted to become LSF floating clients.
- `FLOAT_CLIENTS_ADDR_RANGE=100.*43 100.172.1.13`
  Although one correct address range is specified, because *43 is not correct format, the entire line is considered not valid. An error will be inserted in the LIM log and no hosts will be accepted to become LSF floating clients.

Default Undefined. No security is enabled. Any host in any domain is allowed access to LSF floating client licenses.

HOST_INACTIVITY_LIMIT

Syntax **`HOST_INACTIVITY_LIMIT=`***integer*

Description Integer that is multiplied by EXINTERVAL, the time period you set for the communication between the master and slave LIMs to ensure all parties are functioning.

A slave LIM can send its load information any time from EXINTERVAL to (HOST_INACTIVITY_LIMIT-1)*EXINTERVAL seconds. A master LIM sends a master announce to each host at least every EXINTERVAL*HOST_INACTIVITY_LIMIT seconds.

The HOST_INACTIVITY_LIMIT must be greater than or equal to 2.

Increase or decrease the host inactivity limit to adjust for your tolerance for communication between master and slaves. For example, if you have hosts that frequently become inactive, decrease the host inactivity limit. Note that to get the right interval, you may also have to adjust your EXINTERVAL.

Default 5

LSF_ELIM_BLOCKTIME

Syntax `LSF_ELIM_BLOCKTIME=`*seconds*

Description UNIX only

Maximum amount of time LIM waits for a load update string from the ELIM or MELIM if it is not immediately available.

Use this parameter to add fault-tolerance to LIM when using ELIMs. If there is an error in the ELIM or some situation arises that the ELIM cannot send the entire load update string to the LIM, LIM will not wait indefinitely for load information from ELIM. After the time period specified by LSF_ELIM_BLOCKTIME, the LIM writes the last string sent by ELIM in its log file (`lim.log.`*`host_name`*) and restarts the ELIM.

For example, if LIM is expecting 3 name-value-pairs, such as:

```
3 tmp2 49.5 nio 367.0 licenses 3
```

If after the time period specified by LSF_ELIM_BLOCKTIME LIM has only received the following:

3 tmp2 47.5

LIM writes whatever was received last (`3 tmp2 47.5)` in the log file and restarts the ELIM.

Valid Values Non-negative integers

A value of 0 indicates that LIM will not wait at all to receive information from ELIM—it expects to receive the entire load string at once.

So, if for example, your ELIM writes value-pairs with 1 second intervals between them, and you collect 12 load indices, you need to allow at least 12 seconds for the ELIM to complete writing an entire load string. So you would define LSF_ELIM_BLOCKTIME to 15 or 20 seconds for example.

Default 2 seconds

See Also LSF_ELIM_RESTARTS to limit how many times the ELIM can be restarted.

LSF_ELIM_DEBUG

Syntax `LSF_ELIM_DEBUG=y`

Description UNIX only

This parameter is useful to view which load information an ELIM or MELIM is collecting and to add fault-tolerance to LIM.

When this parameter is set to `y`:

- All load information received by LIM from the ELIM or MELIM is logged in the LIM log file (`lim.log.`*`host_name`*).
- If LSF_ELIM_BLOCKTIME is undefined, whenever there is an error in the ELIM or some situation arises that the ELIM cannot send the entire load update string to the LIM, LIM does not wait indefinitely for load information from ELIM. After 2 seconds, the LIM restarts the ELIM.

For example, LIM is expecting 3 name-value-pairs, such as:

```
3 tmp2 47.5 nio 344.0 licenses 5
```

However, LIM only receives the following from ELIM:

```
3 tmp2 47.5
```

LIM waits 2 seconds after the last value is received and if no more information is received, LIM restarts the ELIM.

If LSF_ELIM_BLOCKTIME is defined, the LIM waits for the specified amount of time before restarting the ELIM instead of the 2 seconds.

Default Undefined—if LSF_ELIM_DEBUG is undefined, load information sent from ELIM to LIM is not logged. In addition, if LSF_ELIM_BLOCKTIME is undefined, LIM waits indefinitely to receive load information from ELIM.

See Also LSF_ELIM_BLOCKTIME to configure how long LIM waits before restarting the ELIM.

LSF_ELIM_RESTARTS to limit how many times the ELIM can be restarted.

LSF_ELIM_RESTARTS

Syntax **LSF_ELIM_RESTARTS=***integer*

Description UNIX only

LSF_ELIM_BLOCKTIME or LSF_ELIM_DEBUG must be defined in conjunction with LSF_ELIM_RESTARTS.

Defines the maximum number of times an ELIM or MELIM can be restarted.

When this parameter is defined:

- If LIM attempts to retrieve load information from the ELIM and there is an error such as an incorrect value, LIM restarts the ELIM.

If the error is consistent and LIM keeps restarting the ELIM, LSF_ELIM_RESTARTS limits how many times the ELIM can be restarted to prevent an ongoing loop.

Valid Values Non-negative integers

Default Undefined; the number of ELIM restarts is unlimited

See Also LSF_ELIM_BLOCKTIME, LSF_ELIM_DEBUG

LSF_HOST_ADDR_RANGE

Syntax **LSF_HOST_ADDR_RANGE=***IP_address ...*

Description Identifies the range of IP addresses that are allowed to be LSF hosts that can be dynamically added to or removed from the cluster.

To enable dynamically added hosts, you must define LSF_HOST_ADDR_RANGE in lsf.cluster.cluster_name, and both LSF_DYNAMIC_HOST_WAIT_TIME and LSF_MASTER_LIST in lsf.conf.

If a value is defined, security for dynamically adding and removing hosts is enabled, and only hosts with IP addresses within the specified range can be added to or removed from a cluster dynamically.

Specify an IP address or range of addresses, in dotted quad notation (`nnn.nnn.nnn.nnn`). Multiple ranges can be defined, separated by spaces.

If there is an error in the configuration of this variable (for example, an address range is not in the correct format), no host will be allowed to join the cluster dynamically and an error message will be logged in the LIM log. Address ranges are validated at configuration time, so they must conform to the required format.

If a requesting host belongs to an IP address that falls in the specified range, the host will be accepted to become a dynamic LSF host.

IP addresses are separated by spaces, and considered "OR" alternatives.

The asterisk (*) character indicates any value is allowed.

The dash (-) character indicates an explicit range of values. For example 1-4 indicates 1,2,3,4 are allowed.

Open ranges such as *-30, or 10-*, are allowed.

If a range is specified with less fields than an IP address such as 10.161, it is considered as 10.161.*.*.

This parameter is limited to 255 characters.

Notes

After you configure LSF_HOST_ADDR_RANGE, check the `lim.log.`*`host_name`* file to make sure this parameter is correctly set. If this parameter is not set or is wrong, this will be indicated in the log file.

Examples

- `LSF_HOST_ADDR_RANGE=100`

  All hosts with a domain address starting with 100 will be allowed access.
- `LSF_HOST_ADDR_RANGE=100-110.34.1-10.4-56`

  All hosts belonging to a domain with an address having the first number between 100 and 110, then 34, then a number between 1 and 10, then, a number between 4 and 56 will be allowed access.
  Example: 100.34.9.45, 100.34.1.4, 102.34.3.20, etc.
- `LSF_HOST_ADDR_RANGE=100.172.1.13 100.*.30-54 124.24-*.1.*-34`

  All hosts belonging to a domain with the address 100.172.1.13 will be allowed access. All hosts belonging to domains starting with 100, then any number, then a range of 30 to 54 will be allowed access. All hosts belonging to domains starting with 124, then from 24 onward, then 1, then from 0 to 34 will be allowed access.
- `LSF_HOST_ADDR_RANGE=12.23.45.*`

  All hosts belonging to domains starting with 12.23.45 are allowed.
- `LSF_HOST_ADDR_RANGE=100.*43`

  The `*` character can only be used to indicate any value. The format of this example is not correct, and an error will be inserted in the LIM log and no hosts will be able to join the cluster dynamically.
- `LSF_HOST_ADDR_RANGE=100.*43 100.172.1.13`

  Although one correct address range is specified, because *43 is not correct format, the entire line is considered not valid. An error will be inserted in the LIM log and no hosts will be able to join the cluster dynamically.

Default *.*.*.*

No security is enabled. Any host in any domain can join the LSF cluster dynamically if you enabled dynamic host configuration.

MASTER_INACTIVITY_LIMIT

Syntax **`MASTER_INACTIVITY_LIMIT=`***integer*

Description An integer reflecting a multiple of EXINTERVAL. A slave will attempt to become master if it does not hear from the previous master after (HOST_INACTIVITY_LIMIT +*host_number**MASTER_INACTIVITY_LIMIT)*EXINTERVAL seconds, where *host_number* is the position of the host in `lsf.cluster.`*`cluster_name`*.

The master host is *host_number* 0.

Default 2

PROBE_TIMEOUT

Syntax **`PROBE_TIMEOUT=`***time_in_seconds*

Description Specifies the timeout in seconds to be used for the `connect`(2) system call

Before taking over as the master, a slave LIM will try to connect to the last known master via TCP.

Default 2 seconds

PRODUCTS

Syntax **`PRODUCTS=`***product_name* ...

Description Specifies the LSF products and features that the cluster will run (you must also have a license for every product you want to run). The list of items is separated by space.

The PRODUCTS parameter is set automatically during installation to include core features. Here are some of the optional products and features that can be specified:

- LSF_Make
- LSF_MultiCluster
- LSF_Sched_Advance_Reservation

Default LSF_Base LSF_Manager LSF_Sched_Fairshare LSF_Sched_Preemption LSF_Sched_Parallel LSF_Sched_Resource_Reservation

RETRY_LIMIT

Syntax **`RETRY_LIMIT=`***integer*

Description Integer reflecting a multiple of EXINTERVAL that controls the number of retries a master or slave LIM makes before assuming that the slave or master is unavailable.

If the master does not hear from a slave for HOST_INACTIVITY_LIMIT exchange intervals, it will actively poll the slave for RETRY_LIMIT exchange intervals before it will declare the slave as unavailable. If a slave does not hear from the master for HOST_INACTIVITY_LIMIT exchange intervals, it will actively poll the master for RETRY_LIMIT intervals before assuming that the master is down.

Default 2

ClusterAdmins Section

(Optional) The `ClusterAdmins` section defines the LSF administrators for the cluster. The only keyword is ADMINISTRATORS.

If the `ClusterAdmins` section is not present, the default LSF administrator is root. Using `root` as the primary LSF administrator is not recommended.

ADMINISTRATORS

Syntax **`ADMINISTRATORS=`***administrator_name ...*

Description Specify UNIX user names.

You can also specify UNIX user group name, Windows user names, and Windows user group names.

The first administrator of the expanded list is considered the primary LSF administrator. The primary administrator is the owner of the LSF configuration files, as well as the working files under `LSB_SHAREDIR/`*`cluster_name`*. If the primary administrator is changed, make sure the owner of the configuration files and the files under `LSB_SHAREDIR/`*`cluster_name`* are changed as well.

Administrators other than the primary LSF administrator have the same privileges as the primary LSF administrator except that they do not have permission to change LSF configuration files. They can perform clusterwide operations on jobs, queues, or hosts in the system.

For flexibility, each cluster may have its own LSF administrators, identified by a user name, although the same administrators can be responsible for several clusters.

Use the `-l` option of the `lsclusters(1)` command to display all of the administrators within a cluster.

Windows domain

- If the specified user or user group is a domain administrator, member of the `Power Users` group or a group with domain administrative privileges, the specified user or user group must belong to the LSF user domain.
- If the specified user or user group is a user or user group with a lower degree of privileges than outlined in the previous point, the user or user group must belong to the LSF user domain and be part of the Global Admins group.

  Windows workgroup

- If the specified user or user group is not a workgroup administrator, member of the `Power Users` group, or a group with administrative privileges on each host, the specified user or user group must belong to the Local Admins group on each host.

Compatibility For backwards compatibility, ClusterManager and Manager are synonyms for `ClusterAdmins` and ADMINISTRATORS respectively. It is possible to have both sections present in the same `lsf.cluster.`*`cluster_name`* file to allow daemons from different LSF versions to share the same file.

Example The following gives an example of a cluster with two LSF administrators. The user listed first, user2, is the primary administrator.

```
Begin ClusterAdmins
ADMINISTRATORS = user2 user7
End ClusterAdmins
```

Default lsfadmin

Host Section

The Host section is the last section in `lsf.cluster.cluster_name` and is the only required section. It lists all the hosts in the cluster and gives configuration information for each host.

The order in which the hosts are listed in this section is important, because the first host listed becomes the LSF master host. Since the master LIM makes all placement decisions for the cluster, it should be on a fast machine.

The LIM on the first host listed becomes the master LIM if this host is up; otherwise, that on the second becomes the master if its host is up, and so on. Also, to avoid the delays involved in switching masters if the first machine goes down, the master should be on a reliable machine. It is desirable to arrange the list such that the first few hosts in the list are always in the same subnet. This avoids a situation where the second host takes over as master when there are communication problems between subnets.

Configuration information is of two types:

- Some fields in a host entry simply describe the machine and its configuration.
- Other fields set thresholds for various resources.

Example Host section

This example Host section contains descriptive and threshold information for three hosts:

```
Begin Host
HOSTNAME    model     type    server r1m pg tmp RESOURCES        RUNWINDOW
hostA       SparcIPC Sparc   1      3.5 15   0 (sunos frame)    ()
hostD       Sparc10  Sparc   1      3.5 15   0 (sunos)          (5:18:30-1:8:30)
hostD       !        !       1      2.0 10   0 ()               ()
hostE       !        !       1      2.0 10   0 (linux !bigmem)  ()
End Host
```

Descriptive fields

The following fields are required in the Host section:

- HOSTNAME
- RESOURCES
- type
- model

The following fields are optional:

- server
- nd
- RUNWINDOW
- REXPRI

HOSTNAME

Description Official name of the host as returned by `hostname`(1)

The name must be listed in `lsf.shared` as belonging to this cluster.

model

Description Host model

The name must be defined in the HostModel section of `lsf.shared`. This determines the CPU speed scaling factor applied in load and placement calculations.

Optionally, the `!` keyword for the model or type column, indicates that the host model or type is to be automatically detected by the LIM running on the host.

nd

Description Number of local disks

This corresponds to the `ndisks` static resource. On most host types, LSF automatically determines the number of disks, and the nd parameter is ignored.

nd should only count local disks with file systems on them. Do not count either disks used only for swapping or disks mounted with NFS.

Default The number of disks determined by the LIM, or 1 if the LIM cannot determine this

RESOURCES

Description The static Boolean resources available on this host

The resource names are strings defined in the Resource section of `lsf.shared`. You may list any number of resources, enclosed in parentheses and separated by blanks or tabs. For example:

```
(fs frame hpux)
```

Optionally, you can specify an exclusive resource by prefixing the resource with an exclamation mark (!). For example, resource `bigmem` is defined in `lsf.shared`, and is defined as an exclusive resource for `hostE`:

```
Begin Host
HOSTNAME   model     type    server r1m pg  tmp RESOURCES        RUNWINDOW
...
hostE      !         !       1      2.0 10   0 (linux !bigmem)   ()
...
End Host
```

You must explicitly specify the exclusive resources in the resource requirements for the job to select a host with an exclusive resource for a job. For example:

```
% bsub -R "bigmem" myjob
```

or

```
% bsub -R "defined(bigmem)" myjob
```

REXPRI

Description UNIX only

Default execution priority for interactive remote jobs run under the RES

The range is from -20 to 20. REXPRI corresponds to the BSD-style nice value used for remote jobs. For hosts with System V-style nice values with the range 0 - 39, a REXPRI of -20 corresponds to a nice value of 0, and +20 corresponds to 39. Higher values of REXPRI correspond to lower execution priority; -20 gives the highest priority, 0 is the default priority for login sessions, and +20 is the lowest priority.

Default 0

RUNWINDOW

Description Dispatch window for interactive tasks.

When the host is not available for remote execution, the host status is `lockW` (locked by run window). LIM does not schedule interactive tasks on hosts locked by dispatch windows. Run windows only apply to interactive tasks placed by LIM. The LSF batch system uses its own (optional) host dispatch windows to control batch job processing on batch server hosts.

Format A dispatch window consists of one or more time windows in the format *begin_time*-*end_time*. No blanks can separate *begin_time* and *end_time*. Time is specified in the form [*day*:]*hour*[:*minute*]. If only one field is specified, LSF assumes it is an *hour*. Two fields are assumed to be *hour*:*minute*. Use blanks to separate time windows.

Default Always accept remote jobs

server

Description Indicates whether the host can receive jobs from other hosts

Specify 1 if the host can receive jobs from other hosts; specify 0 otherwise. Servers that are set to 0 are LSF clients. Client hosts do not run the LSF daemons. Client hosts can submit interactive and batch jobs to the cluster, but they cannot execute jobs sent from other hosts.

Default 1

type

Description Host type as defined in the HostType section of `lsf.shared`

The strings used for host types are determined by the system administrator: for example, SUNSOL, DEC, or HPPA. The host type is used to identify binary-compatible hosts.

The host type is used as the default resource requirement. That is, if no resource requirement is specified in a placement request, the task is run on a host of the same type as the sending host.

Often one host type can be used for many machine models. For example, the host type name SUNSOL6 might be used for any computer with a SPARC processor running SunOS 6. This would include many Sun models and quite a few from other vendors as well.

Optionally, the `!` keyword for the model or type column, indicates that the host model or type is to be automatically detected by the LIM running on the host.

Threshold fields

The LIM uses these thresholds in determining whether to place remote jobs on a host. If one or more LSF load indices exceeds the corresponding threshold (too many users, not enough swap space, etc.), then the host is regarded as busy, and LIM will not recommend jobs to that host.

The CPU run queue length threshold values (r15s, r1m, and r15m) are taken as effective queue lengths as reported by `lsload -E`.

All of these fields are optional; you only need to configure thresholds for load indices that you wish to use for determining whether hosts are busy. Fields that are not configured are not considered when determining host status. The keywords for the threshold fields are not case sensitive.

Thresholds can be set for any of the following:

- The built-in LSF load indexes (`r15s`, `r1m`, `r15m`, `ut`, `pg`, `it`, `io`, `ls`, `swp`, `mem`, `tmp`)
- External load indexes defined in the Resource section of `lsf.shared`

ResourceMap Section

The ResourceMap section defines shared resources in your cluster. This section specifies the mapping between shared resources and their sharing hosts. When you define resources in the Resources section of `lsf.shared`, there is no distinction between a shared and non-shared resource. By default, all resources are not shared and are local to each host. By defining the ResourceMap section, you can define resources that are shared by all hosts in the cluster or define resources that are shared by only some of the hosts in the cluster.

This section must appear after the Host section of `lsf.cluster.`*`cluster_name`*, because it has a dependency on host names defined in the Host section.

ResourceMap section structure

The first line consists of the keywords RESOURCENAME and LOCATION. Subsequent lines describe the hosts that are associated with each configured resource.

Example ResourceMap section

```
Begin ResourceMap
RESOURCENAME   LOCATION
verilog        (5@[all])
local          ([host1 host2] [others])
End ResourceMap
```

The resource `verilog` must already be defined in the RESOURCE section of the `lsf.shared` file. It is a static numeric resource shared by all hosts. The value for `verilog` is 5. The resource `local` is a numeric shared resource that contains two instances in the cluster. The first instance is shared by two machines, `host1` and `host2`. The second instance is shared by all other hosts.

Resources defined in the ResourceMap section can be viewed by using the `-s` option of the `lshosts` (for static resource) and `lsload` (for dynamic resource) commands.

LOCATION

Description

Defines the hosts that share the resource

For a static resource, you must define an initial value here as well. Do not define a value for a dynamic resource.

instance is a list of host names that share an instance of the resource. The reserved words `all`, `others`, and `default` can be specified for the instance:

- `all`—Indicates that there is only one instance of the resource in the whole cluster and that this resource is shared by all of the hosts

  Use the not operator (~) to exclude hosts from the `all` specification. For example:

  `(2@[all ~host3 ~host4])`

  means that 2 units of the resource are shared by all server hosts in the cluster made up of `host1 host2 ...` host*n*, except for `host3` and `host4`. This is useful if you have a large cluster but only want to exclude a few hosts.

  The parentheses are required in the specification. The not operator can only be used with the `all` keyword. It is not valid with the keywords `others` and `default`.

- `others`—Indicates that the rest of the server hosts not explicitly listed in the LOCATION field comprise one instance of the resource

  For example:

  `2@[host1] 4@[others]`

  indicates that there are 2 units of the resource on `host1` and 4 units of the resource shared by all other hosts.

- `default`—Indicates an instance of a resource on each host in the cluster

  This specifies a special case where the resource is in effect not shared and is local to every host. `default` means at each host. Normally, you should not need to use `default,` because by default all resources are local to each host. You might want to use ResourceMap for a non-shared static resource if you need to specify different values for the resource on different hosts.

RESOURCENAME

Description

Name of the resource

This resource name must be defined in the Resource section of `lsf.shared`. You must specify at least a name and description for the resource, using the keywords RESOURCENAME and DESCRIPTION.

- A resource name cannot begin with a number.
- A resource name cannot contain any of the following characters:

  `:  .  (  )  [  +  - *  /  !  &  | <  >  @  =`

- A resource name cannot be any of the following reserved names:

  `cpu cpuf io logins ls idle maxmem maxswp maxtmp type model status it mem ncpus ndisks pg r15m r15s r1m swap swp tmp ut`

- Resource names are case sensitive
- Resource names can be up to 29 characters in length

RemoteClusters Section

Optional. This section is used only in a MultiCluster environment. By default, the local cluster can obtain information about all other clusters specified in `lsf.shared`. The RemoteClusters section limits the clusters that the local cluster can obtain information about.

The RemoteClusters section is required if you want to configure cluster equivalency, cache interval, daemon authentication across clusters, or if you want to run parallel jobs across clusters. To maintain compatibility in this case, make sure the list includes all clusters specified in `lsf.shared`, even if you only configure the default behavior for some of the clusters.

The first line consists of keywords. CLUSTERNAME is mandatory and the other parameters are optional.

Subsequent lines configure the remote cluster.

Example RemoteClusters section

```
Begin RemoteClusters
CLUSTERNAME   EQUIV   CACHE_INTERVAL   RECV_FROM   AUTH
cluster1        Y           60             Y       KRB
cluster2        N           60             Y       -
cluster4        N           60             N       PKI
End RemoteClusters
```

CLUSTERNAME

Description

Remote cluster name

Defines the Remote Cluster list. Specify the clusters you want the local cluster will recognize. Recognized clusters must also be defined in `lsf.shared`. Additional clusters listed in `lsf.shared` but not listed here will be ignored by this cluster.

EQUIV

Description

Specify 'Y' to make the remote cluster equivalent to the local cluster. Otherwise, specify 'N'. The master LIM considers all equivalent clusters when servicing requests from clients for load, host, or placement information.

EQUIV changes the default behavior of LSF commands and utilities and causes them to automatically return load (`lsload`(1)), host (`lshosts`(1)), or placement (`lsplace`(1)) information about the remote cluster as well as the local cluster, even when you don't specify a cluster name.

CACHE_INTERVAL

Description

Specify the load information cache threshold, in seconds. The host information threshold is twice the value of the load information threshold.

To reduce overhead and avoid updating information from remote clusters unnecessarily, LSF displays information in the cache, unless the information in the cache is older than the threshold value.

Default

60 (seconds)

RECV_FROM

Description Specifies whether the local cluster accepts parallel jobs that originate in a remote cluster

RECV_FROM does not affect regular or interactive batch jobs.

Specify 'Y' if you want to run parallel jobs across clusters. Otherwise, specify 'N'.

Default Y

AUTH

Description Defines the preferred authentication method for LSF daemons communicating across clusters. Specify the same method name that is used to identify the corresponding `eauth` program (**`eauth`**.*method_name*). If the remote cluster does not prefer the same method, LSF uses default security between the two clusters.

Default - (only privileged port (setuid) authentication is used between clusters)

lsf.*cluster_name*.license.acct

This is the license accounting file. There is one for each cluster, called `lsf.`*`cluster_name`*`.license.acct`. The *cluster_name* variable is the name of the cluster defined in the Cluster section of `lsf.shared`.

The `lsf.`*`cluster_name`*`.license.acct` file contains two types of configuration information:

- LSF license information
- MultiCluster license information
- Dual-core CPU license information

Contents

- "lsf.cluster_name.license.acct Structure" on page 500

lsf.*cluster_name*.license.acct Structure

The license audit log file is an ASCII file with one record per line. The fields of a record are separated by blanks.

File properties

Location The default location of this file is defined by `LSF_LOGDIR` in `lsf.conf`, but you can override this by defining `LSF_LICENSE_ACCT_PATH` in `lsf.conf`.

Owner The primary LSF admin is the owner of this file.

Permissions `-rw-r--r--`

Records and fields

The fields of a record are separated by blanks. The fields in order of occurrence are as follows:

`timestamp (%d)`

Time stamp of the logged event (in seconds since the epoch).

`type (%s)`

The LSF product type. The valid values are as follows:

- `LSF_MANAGER`
- `LSF_MULTICLUSTER`
- `LSF_DUALCORE`

`version (%s)`

The version of the LSF product.

`value (%s)`

The actual tracked value. The format of this field depends on the product type as specified by the `type` field:

`LSF_MANAGER`

`E` *e_peak e_max_avail* `S` *s_peak s_max_avail* `B` *b_peak b_max_avail*

Where

- *e_peak*, *s_peak*, and *b_peak* are the peak usage values (in number of CPUs) of the E, S, and B class licenses, respectively.
- *e_max_avail*, *s_max_avail*, and *b_max_avail* are the maximum availability and usage values (in number of CPUs) of the E, S, and B class licenses, respectively. This is determined by the license that you purchased.

`LSF_MULTICLUSTER`

mc_peak mc_max_avail

Where

- *mc_peak* is the peak usage value (in number of CPUs) of the LSF MultiCluster license
- *mc_max_avail* is the maximum availability and usage (in number of CPUs) of the LSF MultiCluster license. This is determined by the license that you purchased.

`LSF_DUALCORE`

mc_peak mc_max_avail

Where

- *mc_peak* is the peak usage value (in number of CPUs) of the LSF dual-core CPU license
- *mc_max_avail* is the maximum availability and usage (in number of CPUs) of the LSF dual-core CPU license. This is determined by the license that you purchased.

`status (%s)`

The results of the license usage check. The valid values are as follows:

- `OK`

  Peak usage is less than the maximum license availability
- `OVERUSE`

  Peak usage is more than the maximum license availability

`hash (%s)`

Line encryption used to authenticate the record.

Example record Format

```
1128372131 LSF_MANAGER 6.2 E hostA OVERUSE 7c7998a6861ea119cd48414a820be18cd641
1128372131 LSF_DUALCORE_X86 6.2 0 2 OK eda7876ed6da4f15286a08b404cd4b37f98f9c6e
1128372131 LSF_MULTICLUSTER 6.2 8 10 OK 281288c606a50065ea0e2f3e7161972c56491dc
1128372185 LSF_MANAGER 6.2 E 8 0 S 0 2 B 0 10 OVERUSE fb439ee293821761af9ed0785
1128372185 LSF_MANAGER 6.2 E hostA OVERUSE 2d22a06d6c5cfd5aba40875c2cb8544444a5
```

SEE ALSO

LSF_LOGDIR in `lsf.conf`

LSF_LICENSE_ACCT_PATH in `lsf.conf`

lsf.conf

Contents

- "About lsf.conf" on page 504
- "Parameters" on page 505

About lsf.conf

The lsf.conf file controls the operation of LSF.

The lsf.conf file is created during installation by the LSF setup program, and records all the settings chosen when LSF was installed. The lsf.conf file dictates the location of the specific configuration files and operation of individual servers and applications.

The lsf.conf file is used by LSF and applications built on top of it. For example, information in lsf.conf is used by LSF daemons and commands to locate other configuration files, executables, and network services. lsf.conf is updated, if necessary, when you upgrade to a new version.

This file can also be expanded to include application-specific parameters.

Changing lsf.conf configuration

After making any changes to lsf.conf, run the following commands:

- lsadmin reconfig to reconfigure LIM
- badmin mbdrestart to restart mbatchd

Location

The default location of lsf.conf is in /etc. This default location can be overridden when necessary by either the environment variable LSF_ENVDIR or the command line option -d available to some of the applications.

Format

Each entry in lsf.conf has one of the following forms:

```
NAME=VALUE
NAME=
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each NAME even if no value follows and there should be no space beside the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Lines starting with a pound sign (#) are comments and are ignored. Do not use #if as this is reserved syntax for time-based configuration.

Parameters

- LSB_API_CONNTIMEOUT
- LSB_API_RECVTIMEOUT
- LSB_CPUSET_BESTCPUS
- LSB_BLOCK_JOBINFO_TIMEOUT
- LSB_CHUNK_RUSAGE
- LSB_CMD_LOG_MASK
- LSB_CMD_LOGDIR
- LSB_CONFDIR
- LSB_CRDIR
- LSB_DEBUG
- LSB_DEBUG_CMD
- LSB_DEBUG_MBD
- LSB_DEBUG_NQS
- LSB_DEBUG_SBD
- LSB_DEBUG_SCH
- LSB_DEFAULT_PJLTYPE
- LSB_DISABLE_RERUN_POST_EXEC
- LSB_ECHKPNT_KEEP_OUTPUT
- LSB_ECHKPNT_METHOD
- LSB_ECHKPNT_METHOD_DIR
- LSB_ESUB_METHOD
- LSB_INTERACT_MSG_ENH
- LSB_INTERACT_MSG_INTVAL
- LSB_IRIX_NODESIZE (OBSOLETE)
- LSB_KEEP_SYSDEF_RLIMIT
- LSB_JOB_CPULIMIT
- LSB_JOB_MEMLIMIT
- LSB_LOCALDIR
- LSB_MAILPROG
- LSB_MAILSERVER
- LSB_MAILSIZE_LIMIT
- LSB_MAILTO
- LSB_MAX_JOB_DISPATCH_PER_SESSION
- LSB_MAX_PROBE_SBD
- LSB_MAX_NQS_QUEUES
- LSB_MBD_PORT
- LSB_MC_CHKPNT_RERUN
- LSB_MC_INITFAIL_MAIL
- LSB_MC_INITFAIL_RETRY
- LSB_MEMLIMIT_ENFORCE

- LSB_MIG2PEND
- LSB_MOD_ALL_JOBS
- LSB_NCPU_ENFORCE
- LSB_NQS_PORT
- LSB_PSET_BIND_DEFAULT
- LSB_QUERY_PORT
- LSB_REQUEUE_TO_BOTTOM
- LSB_RLA_HOST_LIST
- LSB_RLA_PORT
- LSB_RLA_UPDATE
- LSB_RLA_WORKDIR
- LSB_RMSACCT_DELAY
- LSB_RMS_MAXNUMNODES
- LSB_RMS_MAXNUMRAILS
- LSB_RMS_MAXPTILE
- LSB_SLURM_BESTFIT
- LSB_SBD_PORT
- LSB_SET_TMPDIR
- LSB_SHAREDIR
- LSB_SHORT_HOSTLIST
- LSB_SIGSTOP
- LSB_SUB_COMMANDNAME
- LSB_STDOUT_DIRECT
- LSB_TIME_CMD
- LSB_TIME_MBD
- LSB_TIME_RESERVE_NUMJOBS
- LSB_TIME_SBD
- LSB_TIME_SCH
- LSB_UTMP
- LSF_AFS_CELLNAME
- LSF_AM_OPTIONS
- LSF_API_CONNTIMEOUT
- LSF_API_RECVTIMEOUT
- LSF_AUTH
- LSF_AUTH_DAEMONS
- LSF_BINDIR
- LSF_CMD_LOGDIR
- LSF_CMD_LOG_MASK
- LSF_CONF_RETRY_INT
- LSF_CONF_RETRY_MAX
- LSF_CONFDIR
- LSF_DAEMON_WRAP

- LSF_DEBUG_LIM
- LSF_DEBUG_RES
- LSF_DHCP_ENV
- LSF_DISPATCHER_LOGDIR
- LSF_DYNAMIC_HOST_WAIT_TIME
- LSF_ENABLE_CSA
- LSF_ENABLE_DUALCORE
- LSF_ENABLE_EXTSCHEDULER
- LSF_ENVDIR
- LSF_EVENT_PROGRAM
- LSF_EVENT_RECEIVER
- LSF_HPC_EXTENSIONS
- LSF_HPC_NCPU_COND
- LSF_HPC_NCPU_INCREMENT
- LSF_HPC_NCPU_INCR_CYCLES
- LSF_HPC_NCPU_THRESHOLD
- LSF_HPC_PJL_LOADENV_TIMEOUT
- LSF_ID_PORT
- LSF_INCLUDEDIR
- LSF_INDEP
- LSF_INTERACTIVE_STDERR
- LSF_IRIX_BESTCPUS (OBSOLETE)
- LSF_LD_SECURITY
- LSF_LIBDIR
- LSF_LIC_SCHED_HOSTS
- LSF_LIC_SCHED_PREEMPT_REQUEUE
- LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE
- LSF_LIC_SCHED_PREEMPT_STOP
- LSF_LICENSE_ACCT_PATH
- LSF_LICENSE_FILE
- LSF_LICENSE_NOTIFICATION_INTERVAL
- LSF_LIM_DEBUG
- LSF_LIM_IGNORE_CHECKSUM
- LSF_LIM_PLUGINDIR
- LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT
- LSF_LIM_SOL27_PLUGINDIR
- LSF_LOCAL_RESOURCES
- LSF_LOG_MASK
- LSF_LOG_MASK_WIN
- LSF_LOGDIR
- LSF_LOGDIR_USE_WIN_REG
- LSF_MACHDEP

- LSF_MANDIR
- LSF_MASTER_LIST
- LSF_MC_NON_PRIVILEGED_PORTS
- LSF_MISC
- LSF_NON_PRIVILEGED_PORTS
- LSF_NIOS_DEBUG
- LSF_NIOS_JOBSTATUS_INTERVAL
- LSF_NIOS_RES_HEARTBEAT
- LSF_PAM_HOSTLIST_USE
- LSF_PAM_PLUGINDIR
- LSF_PAM_USE_ASH
- LSF_POE_TIMEOUT_BIND
- LSF_POE_TIMEOUT_SELECT
- LSF_PIM_INFODIR
- LSF_PIM_SLEEPTIME
- LSF_PIM_SLEEPTIME_UPDATE
- LSF_RES_ACCT
- LSF_RES_ACCTDIR
- LSF_RES_ACTIVE_TIME
- LSF_RES_CONNECT_RETRY
- LSF_RES_DEBUG
- LSF_RES_PLUGINDIR
- LSF_RES_PORT
- LSF_RES_RLIMIT_UNLIM
- LSF_RES_SOL27_PLUGINDIR
- LSF_RES_TIMEOUT
- LSF_ROOT_REX
- LSF_RSH
- LSF_SECUREDIR
- LSF_SERVER_HOSTS
- LSF_SERVERDIR
- LSF_SHELL_AT_USERS
- LSF_SHIFT_JIS_INPUT
- LSF_SLURM_DISABLE_CLEANUP
- LSF_SLURM_TMPDIR
- LSF_STRICT_CHECKING
- LSF_STRIP_DOMAIN
- LSF_TIME_CMD
- LSF_TIME_LIM
- LSF_TIME_RES
- LSF_TMPDIR
- LSF_TOPD_PORT

- LSF_TOPD_WORKDIR
- LSF_ULDB_DOMAIN
- LSF_USE_HOSTEQUIV
- LSF_USER_DOMAIN
- LSF_VPLUGIN
- MC_PLUGIN_REMOTE_RESOURCE
- XLSF_APPDIR
- XLSF_UIDDIR

LSB_API_CONNTIMEOUT

Syntax **LSB_API_CONNTIMEOUT=***time_seconds*

Description The timeout in seconds when connecting to LSF.

Valid Values Any positive integer or zero

Default 10

See also LSB_API_RECVTIMEOUT

LSB_API_RECVTIMEOUT

Syntax **LSB_API_RECVTIMEOUT=***time_seconds*

Description Timeout in seconds when waiting for a reply from LSF.

Valid values Any positive integer or zero

Default 10

See also LSB_API_CONNTIMEOUT

LSB_CPUSET_BESTCPUS

Syntax **LSB_CPUSET_BESTCPUS=y** | **Y**

Description If set, enables the best-fit algorithm for SGI cpusets

LSF_IRIX_BESTCPUS is obsolete.

Default Y (best-fit)

LSB_BLOCK_JOBINFO_TIMEOUT

Syntax **LSB_BLOCK_JOBINFO_TIMEOUT=***time_minutes*

Description Timeout in minutes for job information query commands (e.g., `bjobs`).

Valid values Any positive integer

Default Undefined (no timeout)

See also MAX_JOBINFO_QUERY_PERIOD in “lsb.params”

LSB_CHUNK_RUSAGE

Syntax **LSB_CHUNK_RUSAGE=y**

Description Applies only to chunk jobs. When set, sbatchd contacts PIM to retrieve resource usage information to enforce resource usage limits on chunk jobs.

By default, resource usage limits are not enforced for chunk jobs because chunk jobs are typically too short to allow LSF to collect resource usage.

If LSB_CHUNK_RUSAGE=Y is defined, limits may not be enforced for chunk jobs that take less than a minute to run.

Default Undefined. No resource usage is collected for chunk jobs.

LSB_CMD_LOG_MASK

Syntax **LSB_CMD_LOG_MASK=***log_level*

Description Specifies the logging level of error messages from LSF commands.

To specify the logging level of error messages for LSF commands, use LSF_CMD_LOG_MASK. To specify the logging level of error messages for LSF daemons, use LSF_LOG_MASK.

LSB_CMD_LOG_MASK sets the log level and is used in combination with LSB_DEBUG_CMD, which sets the log class for LSF batch commands. For example:

```
LSB_CMD_LOG_MASK=LOG_DEBUG
LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

LSF commands log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSB_CMD_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

The commands log to the syslog facility unless LSB_CMD_LOGDIR is set.

Valid values The log levels from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2

- LOG_DEBUG3

Default LOG_WARNING

See also LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSB_CMD_LOGDIR

Syntax **LSB_CMD_LOGDIR=***path*

Description Specifies the path to the LSF command log files.

Default `/tmp`

See also LSB_CMD_LOG_MASK, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSB_CONFDIR

Syntax **LSB_CONFDIR=***path*

Description Specifies the path to the directory containing the LSF configuration files.

The configuration directories are installed under LSB_CONFDIR.

Configuration files for each cluster are stored in a subdirectory of LSB_CONFDIR. This subdirectory contains several files that define user and host lists, operation parameters, and queues.

All files and directories under LSB_CONFDIR must be readable from all hosts in the cluster. `LSB_CONFDIR/`*`cluster_name`*`/configdir` must be owned by the LSF administrator.

CAUTION **Do not change this parameter after LSF has been installed.**

Default `LSF_CONFDIR/lsbatch`

See also LSF_CONFDIR

LSB_CRDIR

Syntax **LSB_CRDIR=***path*

Description Specifies the path and directory to the checkpointing executables on systems that support kernel-level checkpointing. LSB_CRDIR specifies the directory containing the `chkpnt` and `restart` utility programs that `sbatchd` uses to checkpoint or restart a job.

For example:

```
LSB_CRDIR=/usr/bin
```

If your platform supports kernel-level checkpointing, and if you want to use the utility programs provided for kernel-level checkpointing, set LSB_CRDIR to the location of the utility programs.

Default Undefined

If undefined, the system uses /bin.

LSB_DEBUG

Syntax **LSB_DEBUG=1 | 2**

Description Sets the LSF batch system to debug.

If defined, LSF runs in single user mode:

- No security checking is performed
- Daemons do not run as root

When LSB_DEBUG is defined, LSF will not look in the system services database for port numbers. Instead, it uses the port numbers defined by the parameters LSB_MBD_PORT/LSB_SBD_PORT in lsf.conf. If these parameters are not defined, it uses port number 40000 for mbatchd and port number 40001 for sbatchd.

You should always specify 1 for this parameter unless you are testing LSF.

Can also be defined from the command line.

Valid values

- LSB_DEBUG=1
  The LSF system runs in the background with no associated control terminal.
- LSB_DEBUG=2
  The LSF system runs in the foreground and prints error messages to tty.

Default Undefined

See also LSB_DEBUG, LSB_DEBUG_CMD, LSB_DEBUG_MBD, LSB_DEBUG_NQS, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSB_DEBUG_CMD

Syntax **LSB_DEBUG_CMD=***log_class*

Description Sets the debugging log class for commands and APIs.

Specifies the log class filtering that will be applied to LSF batch commands or the API. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_CMD sets the log class and is used in combination with LSB_CMD_LOG_MASK, which sets the log level. For example:

```
LSB_CMD_LOG_MASK=LOG_DEBUG
LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

The daemons log to the syslog facility unless LSB_CMD_LOGDIR is defined.

To specify multiple log classes, use a space-separated list enclosed by quotation marks. For example:

```
LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Can also be defined from the command line.

Valid values Valid log classes are:

- LC_AFS - Log AFS messages
- LC_AUTH - Log authentication messages
- LC_CHKPNT - Log checkpointing messages
- LC_COMM - Log communication messages
- LC_DCE - Log messages pertaining to DCE support
- LC_EEVENTD - Log eeventd messages
- LC_ELIM - Log ELIM messages
- LC_EXEC - Log significant steps for job execution
- LC_FAIR - Log fairshare policy messages
- LC_FILE - Log file transfer messages
- LC_HANG - Mark where a program might hang
- LC_JARRAY - Log job array messages
- LC_JLIMIT - Log job slot limit messages
- LC_LICENCE - Log license management messages
- LC_LOADINDX - Log load index messages
- LC_M_LOG - Log multievent logging messages
- LC_MPI - Log MPI messages
- LC_MULTI - Log messages pertaining to MultiCluster
- LC_PEND - Log messages related to job pending reasons
- LC_PERFM - Log performance messages
- LC_PIM - Log PIM messages
- LC_PREEMPT - Log preemption policy messages
- LC_SIGNAL - Log messages pertaining to signals
- LC_SYS - Log system call messages
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

Default Undefined

See also LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_MBD, LSB_DEBUG_NQS, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSB_DEBUG_MBD

Syntax **LSB_DEBUG_MBD=***log_class*

Description Sets the debugging log class for `mbatchd`.

Specifies the log class filtering that will be applied to `mbatchd`. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_MBD sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSB_DEBUG_MBD="LC_TRACE LC_EXEC"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_MBD="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_MBD for your changes to take effect.

If you use the command `badmin mbddebug` to temporarily change this parameter without changing `lsf.conf`, you will not need to restart the daemons.

Valid Values Valid log classes are the same as for LSB_DEBUG_CMD except for the log classes LC_ELIM and LC_JARRAY which cannot be used with LSB_DEBUG_MBD. See "LSB_DEBUG_CMD" on page 512.

Default Undefined

See also LSF_LOG_MASK, LSF_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_DEBUG_MBD, LSB_DEBUG_NQS, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG, `badmin mbddebug`

LSB_DEBUG_NQS

Syntax **LSB_DEBUG_NQS=***log_class*

Description Sets the log class for debugging the NQS interface.

Specifies the log class filtering that will be applied to NQS. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_NQS sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSB_DEBUG_NQS="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_NQS="LC_TRACE LC_EXEC"
```

This parameter can also be defined from the command line.

Valid values For a list of valid log classes, see "LSB_DEBUG_CMD" on page 512.

Default Undefined

See also LSB_DEBUG_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSB_DEBUG_SBD

Syntax **LSB_DEBUG_SBD=***log_class*

Description Sets the debugging log class for `sbatchd`.

Specifies the log class filtering that will be applied to `sbatchd`. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_SBD sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSB_DEBUG_SBD="LC_TRACE LC_EXEC"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_SBD="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_SBD for your changes to take effect.

If you use the command `badmin sbddebug` to temporarily change this parameter without changing `lsf.conf`, you will not need to restart the daemons.

Valid values Valid log classes are the same as for LSB_DEBUG_CMD except for the log classes LC_ELIM and LC_JARRAY which cannot be used with LSB_DEBUG_SBD. See "LSB_DEBUG_CMD" on page 512.

Default Undefined

See also LSB_DEBUG_MBD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, `badmin`

LSB_DEBUG_SCH

Syntax **`LSB_DEBUG_SCH=`***log_class*

Description Sets the debugging log class for `mbschd`.

Specifies the log class filtering that will be applied to `mbschd`. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_SCH sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSB_DEBUG_SCH="LC_SCHED"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_SCH="LC_SCHED LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_SCH for your changes to take effect.

Valid Values Valid log classes are the same as for LSB_DEBUG_CMD except for the log classes LC_ELIM and LC_JARRAY which cannot be used with LSB_DEBUG_SCH, and LC_HPC, which is only valid for LSB_DEBUG_SCH. See "LSB_DEBUG_CMD" on page 512.

Default Undefined

See also LSB_DEBUG_MBD, LSB_DEBUG_SBD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, `badmin`

LSB_DEFAULT_PJLTYPE

Syntax **LSB_DEFAULT_PJLTYPE="***pjl_type* [*pjl_type*] ...**"**

Description Contains the list of all PJL types as Boolean resources that are intended to be autodetected.

The order of MPI types in the list defines the preference of one type over another within the same host group, but not the host group order itself.

For heterogeneous HPC environments, bsub -a auto recognizes the actual PJL type at execution time. esub.auto checks the value of LSB_DEFAULT_PJLTYPE in lsf.conf.

esub.auto creates an additional -R 'same[]' clause in the job submission string. All PJL types that are listed in the LSB_DEFAULT_PJLTYPE variable are added to the clause, preserving the original order.

The job is scheduled according to its requirements and the additional same[] clause. Each host in the cluster is assigned one or more Boolean resources that correspond to the PJL types (MPI implementations) supported by the host. The hosts are broken into the host groups according to LSF policies. When a suitable group is found, the job is dispatched for execution.

mpirun.lsf determines the appropriate PJL and assigns the value to LSF_PJL_TYPE, based on the values of the Boolean resources assigned to a chosen host group and the preference order specified in LSB_DEFAULT_PJLTYPE.

The Boolean resources for MPI autodection are defined in `lsf.shared` at installation. For example:

```
Begin Resource
RESOURCENAME  TYPE    INTERVAL INCREASING  DESCRIPTION
...
lammpi        Boolean   ()         ()       (LAM MPI)
mpich_gm      Boolean   ()         ()       (MPICH GM MPI)
mpichp4       Boolean   ()         ()       (MPICH P4 MPI)
mvapich       Boolean   ()         ()       (Infiniband MPI)
ibmmpi        Boolean   ()         ()       (IBM POE MPI)
hpmpi         Boolean   ()         ()       (HP MPI)
sgimpi        Boolean   ()         ()       (SGI MPI)
intelmpi      Boolean   ()         ()       (Intel MPI)
...
End Resource
```

You must assign appropriate Boolean resources for MPI type to each host in the Host section of `lsf.cluster.`*`cluster_name`*. For example:

```
Begin   Host
HOSTNAME  model type server r1m  mem  swp  RESOURCES
Linux-01       !    !      1 3.5  ()   ()   (mpich_gm lammpi)
Linux-02       !    !      1 3.5  ()   ()   (mpich_gm lammpi)
Linux-03       !    !      1 3.5  ()   ()   (lammpi)
sierraA        !    !      1 3.5  ()   ()   (rms)
sierraB        !    !      1 3.5  ()   ()   (rms)
End     Host
```

Example `LSB_DEFAULT_PJLTYPE="mpich_gm lammpi rms"`

Default Undefined (no default PJL type)

LSB_DISABLE_RERUN_POST_EXEC

Syntax `LSB_DISABLE_RERUN_POST_EXEC=y | Y`

Description If set, and the job is rerunnable, the POST_EXEC configured in the queue is not executed if the job is rerun.

Running of post-execution commands upon restart of a rerunanble job may not always be desirable. For example, if the post-exec removes certain files, or does other cleanup that should only happen if the job finishes successfully, use LSB_DISABLE_RERUN_POST_EXEC to prevent the post-exec from running and allow the succesful continuation of the job when it reruns.

Default Undefined

LSB_ECHKPNT_KEEP_OUTPUT

Syntax `LSB_ECHKPNT_KEEP_OUTPUT=y | Y`

Description Saves the standard output and standard error of custom `echkpnt` and `erestart` methods to:

- `checkpoint_dir/$LSB_JOBID/echkpnt.out`
- `checkpoint_dir/$LSB_JOBID/echkpnt.err`
- `checkpoint_dir/$LSB_JOBID/erestart.out`
- `checkpoint_dir/$LSB_JOBID/erestart.err`

Can also be defined as an environment variable.

Default Undefined; standard error and standard output messages from custom `echkpnt` and `erestart` programs is directed to `/dev/null` and discarded by LSF.

See also LSB_ECHKPNT_METHOD, LSB_ECHKPNT_METHOD_DIR

LSB_ECHKPNT_METHOD

Syntax **LSB_ECHKPNT_METHOD="***method_name* [*method_name*] ...**"**

Description Name of custom `echkpnt` and `erestart` methods.

Can also be defined as an environment variable, or specified through the `bsub -k` option.

The name you specify here will be used for both your custom `echkpnt` and `erestart` programs. You must assign your custom `echkpnt` and *`erestart`* programs the name `echkpnt.method_name` and `erestart.`*`method_name`*. The programs `echkpnt.method_name` and `erestart.`*`method_name`*. must be in LSF_SERVERDIR or in the directory specified by LSB_ECHKPNT_METHOD_DIR.

Do not define `LSB_ECHKPNT_METHOD=default` as `default` is a reserved keyword to indicate to use LSF's default `echkpnt` and `erestart` methods. You can however, specify `bsub -k "my_dir method=default" my_job` to indicate that you want to use LSF's default checkpoint and restart methods.

When this parameter is undefined in lsf.conf or as an environment variable and no custom method is specified at job submission through bsub -k, LSF uses echkpnt.default and erestart.default to checkpoint and restart jobs.

When this parameter is defined, LSF uses the custom checkpoint and restart methods specified.

Limitations The method name and directory (LSB_ECHKPNT_METHOD_DIR) combination must be unique in the cluster.

For example, you may have two echkpnt applications with the same name such as echkpnt.mymethod but what differentiates them is the different directories defined with LSB_ECHKPNT_METHOD_DIR. It is the cluster administrator's responsibility to ensure that method name and method directory combinations are unique in the cluster.

Default Undefined; LSF uses echkpnt.default and erestart.default to checkpoint and restart jobs

See also LSB_ECHKPNT_METHOD_DIR, LSB_ECHKPNT_KEEP_OUTPUT

LSB_ECHKPNT_METHOD_DIR

Syntax **LSB_ECHKPNT_METHOD_DIR=***path*

Description Absolute path name of the directory in which custom echkpnt and erestart programs are located.

The checkpoint method directory should be accessible by all users who need to run the custom echkpnt and erestart programs.

Can also be defined as an environment variable.

Default Undefined; LSF searches in LSF_SERVERDIR for custom echkpnt and erestart programs

See also LSB_ECHKPNT_METHOD, LSB_ECHKPNT_KEEP_OUTPUT

LSB_ESUB_METHOD

Specifies a mandatory esub method that applies to all job submissions. LSB_ESUB_METHOD lists the names of the esub methods used in addition to any methods specified in the bsub -a option.

For example, LSB_ESUB_METHOD="dce fluent" defines DCE as the mandatory security system, and FLUENT as the mandatory application used on all jobs.

Syntax **LSB_ESUB_METHOD="***method_name* [*method_name*] ...**"**

Description Specifies the name of the mandatory esub method. Can also be defined as an environment variable.

The master esub (mesub) uses the name you specify to invoke the appropriate esub program. The esub and esub.xxx programs must be located in LSF_SERVERDIR.

When this parameter is defined, LSF uses the specified esub method, where *method_name* is one of:

- openmp or pvm—for OpenMP or PVM job submission; esub calls esub.openmp or esub.pvm

- poe—for POE job submission; esub calls esub.poe
- ls_dyna—for LS-Dyna job submission; esub calls esub.ls_dyna
- fluent—for FLUENT job submission; esub calls esub.fluent
- afs or dce—for AFS or DCE security; esub calls esub.afs or esub.dce
- lammpi or mpich_gm—for LAM/MPI or MPI-GM job submission; esub calls esub.lammpi or esub.mpich_gm
- Any other application-specific esub program you provide

The master esub program (LSF_SERVERDIR/mesub) handles job submission requirements of the applications. Application-specific esub programs can specify their own job submission requirements. The value of LSB_ESUB_METHOD is set in the LSB_SUB_ADDITIONAL option in the LSB_SUB_PARM file used by esub.

The value of LSB_ESUB_METHOD is passed to esub, but it does not directly affect the other bsub parameters or behavior. The value of LSB_ESUB_METHOD must correspond to an actual esub file. For example, to use LSB_ESUB_METHOD=fluent, the file esub.fluent must exist in LSF_SERVERDIR.

The name of the esub program must be a valid file name. It can contain only alphanumeric characters, underscore (_) and hyphen (-).

Example LSB_ESUB_METHOD="dce fluent" defines DCE as the mandatory security system, and FLUENT as the mandatory application used on all jobs.

Limitations LSF does not detect conflicting method specifications. For example, you can specify *either* openmp or pvm, but not both. If LSB_ESUB_METHOD="openmp" and bsub -a pvm is specified at job submission, the job may fail or be rejected.

If multiple esub methods are specified, and the return value is LSB_SUB_ABORT_VALUE, esub exits without running the remaining esub methods and returns LSB_SUB_ABORT_VALUE.

Default Undefined

LSB_INTERACT_MSG_ENH

Syntax **LSB_INTERACT_MSG_ENH=y | Y**

Description If set, enables enhanced messaging for interactive batch jobs. To disable interactive batch job messages, set LSB_INTERACT_MSG_ENH to any value other than y or Y; for example, LSB_INTERACT_MSG_ENH=N.

Default Undefined

See also LSB_INTERACT_MSG_INTVAL

LSB_INTERACT_MSG_INTVAL

Syntax **LSB_INTERACT_MSG_INTVAL=***time_seconds*

Description Specifies the update interval in seconds for interactive batch job messages. LSB_INTERACT_MSG_INTVAL is ignored if LSB_INTERACT_MSG_ENH is not set.

Job information that LSF uses to get the pending or suspension reason is updated according to the value of PEND_REASON_UPDATE_INTERVAL in lsb.params.

Default Undefined. If LSB_INTERACT_MSG_INTVAL is set to an incorrect value, the default update interval is 60 seconds.

See also LSB_INTERACT_MSG_ENH

LSB_IRIX_NODESIZE (OBSOLETE)

LSB_IRIX_NODESIZE is obsolete. It is ignored if set.

LSB_KEEP_SYSDEF_RLIMIT

Syntax **LSB_KEEP_SYSDEF_RLIMIT=y | n**

Description If resource limits are configured for a user in the SGI IRIX User Limits Database (ULDB) domain specified in LSF_ULDB_DOMAIN, and there is no domain default, the system default is honored.

If LSF_KEEP_SYSDEF_RLIMIT=n, and no resource limits are configured in the domain for the user and there is no domain default, LSF overrides the system default and sets system limits to unlimited.

Default 90 seconds

LSB_JOB_CPULIMIT

Syntax **LSB_JOB_CPULIMIT=y | n**

Description Determines whether the CPU limit is a per-process limit enforced by the OS or whether it is a per-job limit enforced by LSF:

- The per-process limit is enforced by the OS when the CPU time of one process of the job exceeds the CPU limit.
- The per-job limit is enforced by LSF when the total CPU time of all processes of the job exceed the CPU limit.

This parameter applies to CPU limits set when a job is submitted with `bsub -c`, and to CPU limits set for queues by CPULIMIT in `lsb.queues`.

The setting of LSB_JOB_CPULIMIT has the following effect on how the limit is enforced:

When LSB_JOB_CPULIMIT is	LSF-enforced per-job limit	OS-enforced per-process limit
y	Enabled	Disabled
n	Disabled	Enabled
undefined	Enabled	Enabled

- LSF-enforced per-job limit—When the sum of the CPU time of all processes of a job exceed the CPU limit, LSF sends a SIGXCPU signal (where supported by the operating system) from the operating system to all processes belonging to the job, then SIGINT, SIGTERM and SIGKILL. The interval between signals is 10 seconds by default. The time interval between SIGXCPU, SIGINT, SIGKILL, SIGTERM can be configured with the parameter JOB_TERMINATE_INTERVAL in `lsb.params`.

  SIGXCPU is not supported by Windows.

- OS-enforced per process limit—When one process in the job exceeds the CPU limit, the limit is enforced by the operating system. For more details, refer to your operating system documentation for `setrlimit()`.

Default Undefined

Notes To make LSB_JOB_CPULIMIT take effect, use the command `badmin hrestart all` to restart all `sbatchd`s in the cluster.

Changing the default Terminate job control action—You can define a different terminate action in `lsb.queues` with the parameter JOB_CONTROLS if you do not want the job to be killed. For more details on job controls, see *Administering Platform LSF*.

Limitations If a job is running and the parameter is changed, LSF is not able to reset the type of limit enforcement for running jobs.

- If the parameter is changed from per-process limit enforced by the OS to per-job limit enforced by LSF (LSB_JOB_CPULIMIT=n changed to LSB_JOB_CPULIMIT=y), both per-process limit and per-job limit will affect the running job. This means that signals may be sent to the job either when an individual process exceeds the CPU limit or the sum of the CPU time of all processes of the job exceed the limit. A job that is running may be killed by the OS or by LSF.
- If the parameter is changed from per-job limit enforced by LSF to per-process limit enforced by the OS (LSB_JOB_CPULIMIT=y changed to LSB_JOB_CPULIMIT=n), the job will be allowed to run without limits because the per-process limit was previously disabled.

See also `lsb.queues`(5), `bsub`(1), JOB_TERMINATE_INTERVAL in "lsb.params", LSB_MOD_ALL_JOBS

LSB_JOB_MEMLIMIT

Syntax **`LSB_JOB_MEMLIMIT=y | n`**

Description Determines whether the memory limit is a per-process limit enforced by the OS or whether it is a per-job limit enforced by LSF.

- The per-process limit is enforced by the OS when the memory allocated to one process of the job exceeds the memory limit.
- The per-job limit is enforced by LSF when the sum of the memory allocated to all processes of the job exceeds the memory limit.

This parameter applies to memory limits set when a job is submitted with `bsub -M` *`mem_limit`*, and to memory limits set for queues with MEMLIMIT in `lsb.queues`.

The setting of LSB_JOB_MEMLIMIT has the following effect on how the limit is enforced:

When LSB_JOB_MEMLIMIT is	LSF-enforced per-job limit	OS-enforced per-process limit
y	Enabled	Disabled
n or undefined	Disabled	Enabled

- LSF-enforced per-job limit—When the total memory allocated to all processes in the job exceeds the memory limit, LSF sends the following signals to kill the job: SIGINT, SIGTERM, then SIGKILL. The interval between signals is 10 seconds by default.

  On UNIX, the time interval between SIGINT, SIGKILL, SIGTERM can be configured with the parameter JOB_TERMINATE_INTERVAL in `lsb.params`.
- OS-enforced per process limit—When the memory allocated to one process of the job exceeds the memory limit, the operating system enforces the limit. LSF passes the memory limit to the operating system. Some operating systems apply the memory limit to each process, and some do not enforce the memory limit at all.

  OS memory limit enforcement is only available on systems that support `RLIMIT_RSS` for `setrlimit()`.

  The following operating systems do not support the memory limit at the OS level and the job will be allowed to run without a memory limit:

  - Windows
  - Sun Solaris 2.x

Default Undefined; per-process memory limit enforced by the OS; per-job memory limit enforced by LSF disabled

Notes To make LSB_JOB_MEMLIMIT take effect, use the command `badmin hrestart all` to restart all `sbatchd`s in the cluster.

If LSB_JOB_MEMLIMIT is set, it overrides the setting of the parameter LSB_MEMLIMIT_ENFORCE. The parameter LSB_MEMLIMIT_ENFORCE is ignored.

The difference between LSB_JOB_MEMLIMIT set to y and LSB_MEMLIMIT_ENFORCE set to y is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to y, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Changing the default Terminate job control action—You can define a different Terminate action in `lsb.queues` with the parameter JOB_CONTROLS if you do not want the job to be killed. For more details on job controls, see *Administering Platform LSF*.

Limitations If a job is running and the parameter is changed, LSF is not able to reset the type of limit enforcement for running jobs.

- If the parameter is changed from per-process limit enforced by the OS to per-job limit enforced by LSF (LSB_JOB_MEMLIMIT=n or undefined changed to LSB_JOB_MEMLIMIT=y), both per-process limit and per-job limit will affect the running job. This means that signals may be sent to the job either when the memory allocated to an individual process exceeds the memory limit or the sum of memory allocated to all processes of the job exceed the limit. A job that is running may be killed by LSF.
- If the parameter is changed from per-job limit enforced by LSF to per-process limit enforced by the OS (LSB_JOB_MEMLIMIT=y changed to

LSB_JOB_MEMLIMIT=n or undefined), the job will be allowed to run without limits because the per-process limit was previously disabled.

See also LSB_MEMLIMIT_ENFORCE, LSB_MOD_ALL_JOBS, `lsb.queues`(5), `bsub`(1), JOB_TERMINATE_INTERVAL in "lsb.params"

LSB_LOCALDIR

Syntax **`LSB_LOCALDIR=`***path*

Description Enables duplicate logging.

Specify the path to a local directory that exists only on the first LSF master host (the first host configured in `lsf.cluster.`*cluster_name*). LSF puts the primary copies of the event and accounting log files in this directory. LSF puts the duplicates in LSB_SHAREDIR.

Example `LSB_LOCALDIR=/usr/share/lsbatch/loginfo`

Default Undefined

See also LSB_SHAREDIR, EVENT_UPDATE_INTERVAL in "lsb.params"

LSB_MAILPROG

Syntax **`LSB_MAILPROG=`***file_name*

Description Path and file name of the mail program used by LSF to send email. This is the electronic mail program that LSF will use to send system messages to the user. When LSF needs to send email to users it invokes the program defined by LSB_MAILPROG in `lsf.conf`. You can write your own custom mail program and set LSB_MAILPROG to the path where this program is stored.

LSF administrators can set the parameter as part of cluster reconfiguration. Provide the name of any mail program. For your convenience, LSF provides the `sendmail` mail program, which supports the `sendmail` protocol on UNIX.

In a mixed cluster, you can specify different programs for Windows and UNIX. You can set this parameter during installation on Windows. For your convenience, LSF provides the `lsmail.exe` mail program, which supports SMTP and Microsoft Exchange Server protocols on Windows. If `lsmail` is specified, the parameter LSB_MAILSERVER must also be specified.

If you change your mail program, the LSF administrator must restart `sbatchd` on all hosts to retrieve the new value.

UNIX By default, LSF uses `/usr/lib/sendmail` to send email to users. LSF calls LSB_MAILPROG with two arguments; one argument gives the full name of the sender, and the other argument gives the return address for mail.

LSB_MAILPROG must read the body of the mail message from the standard input. The end of the message is marked by end-of-file. Any program or shell script that accepts the arguments and input, and delivers the mail correctly, can be used.

LSB_MAILPROG must be executable by any user.

Windows If LSB_MAILPROG is not defined, no email is sent.

Examples `LSB_MAILPROG=lsmail.exe`

`LSB_MAILPROG=/serverA/tools/lsf/bin/unixhost.exe`

Default `/usr/lib/sendmail` (UNIX)

blank (Windows)

See also LSB_MAILSERVER, LSB_MAILTO

LSB_MAILSERVER

Syntax **`LSB_MAILSERVER=`***mail_protocol*:*mail_server*

Description Part of mail configuration on Windows.

This parameter only applies when `lsmail` is used as the mail program (LSB_MAILPROG=lsmail.exe).Otherwise, it is ignored.

Both *`mail_protocol`* and *`mail_server`* must be indicated.

Set this parameter to either SMTP or Microsoft Exchange protocol (SMTP or EXCHANGE) and specify the name of the host that is the mail server.

This parameter is set during installation of LSF on Windows or is set or modified by the LSF administrator.

If this parameter is modified, the LSF administrator must restart `sbatchd` on all hosts to retrieve the new value.

Examples `LSB_MAILSERVER=EXCHANGE:Host2@company.com`

`LSB_MAILSERVER=SMTP:MailHost`

Default Undefined

See also LSB_MAILPROG

LSB_MAILSIZE_LIMIT

Syntax **`LSB_MAILSIZE_LIMIT=`***email_size_in_KB*

Description Limits the size of the email containing job output information.

The system sends job information such as CPU, process and memory usage, job output, and errors in email to the submitting user account. Some batch jobs can create large amounts of output. To prevent large job output files from interfering with your mail system, use LSB_MAILSIZE_LIMIT to set the maximum size in KB of the email containing the job information. Specify a positive integer.

If the size of the job output email exceeds LSB_MAILSIZE_LIMIT, the output is saved to a file under JOB_SPOOL_DIR or to the default job output directory if JOB_SPOOL_DIR is undefined. The email informs users of where the job output is located.

If the `-o` option of `bsub` is used, the size of the job output is not checked against LSB_MAILSIZE_LIMIT.

If you use a custom mail program specified by the LSB_MAILPROG parameter that can use the LSB_MAILSIZE environment variable, it is not necessary to configure LSB_MAILSIZE_LIMIT.

Default By default, LSB_MAILSIZE_LIMIT is not enabled. No limit is set on size of batch job output email.

See also LSB_MAILPROG, LSB_MAILTO

LSB_MAILTO

Syntax **LSB_MAILTO=***mail_account*

Description LSF sends electronic mail to users when their jobs complete or have errors, and to the LSF administrator in the case of critical errors in the LSF system. The default is to send mail to the user who submitted the job, on the host on which the daemon is running; this assumes that your electronic mail system forwards messages to a central mailbox.

The LSB_MAILTO parameter changes the mailing address used by LSF. LSB_MAILTO is a format string that is used to build the mailing address.

Common formats are:

- `!U`—Mail is sent to the submitting user's account name on the local host. The substring `!U`, if found, is replaced with the user's account name.
- `!U@company_name.com`—Mail is sent to `user@company_name.com` on the mail server.

  The mail server is specified by LSB_MAILSERVER.
- `!U@!H`—Mail is sent to *`user@submission_hostname`*. The substring `!H` is replaced with the name of the submission host.

  This format is valid on UNIX only. It is not supported on Windows.

All other characters (including any other '`!`') are copied exactly.

If this parameter is modified, the LSF administrator must restart `sbatchd` on all hosts to retrieve the new value.

Default `!U`

See also LSB_MAILPROG, LSB_MAILSIZE_LIMIT

LSB_MAX_JOB_DISPATCH_PER_SESSION

Syntax **LSB_MAX_JOB_DISPATCH_PER_SESSION**=*integer*

Description Defines the maximum number of jobs that mbatchd can dispatch during one job scheduling session.

Both `mbatchd` and `sbatchd` must be restarted when you change the value of this parameter.

If set to a value greater than 300, the file descriptor limit is increased on operating systems that support a file descriptor limit greater than 1024.

Use together with MAX_SBD_CONNS in `lsb.params`. Set MAX_SBD_CONNS to the same value as LSB_MAX_JOB_DISPATCH_PER_SESSION.

Examples

- `LSB_MAX_JOB_DISPATCH_PER_SESSION=300`

  The file descriptor limit is 1024.
- `LSB_MAX_JOB_DISPATCH_PER_SESSION=1000`

  The file descriptor limit is greater than 1024 on operating systems that support a greater limit.

Default 300

See also MAX_SBD_CONNS in "lsb.params"

LSB_MAX_PROBE_SBD

Syntax **LSB_MAX_PROBE_SBD=***integer*

Description Specifies the maximum number of sbatchd instances can be polled by mbatchd in the interval MBD_SLEEP_TIME/10. Use this parameter in large clusters to reduce the time it takes for mbatchd to probe all sbatchds.

The value of LSB_MAX_PROBE_SBD cannot be greater than the number of hosts in the cluster. If it is, mbatchd adjusts the value of LSB_MAX_PROBE_SBD to be same as the number of hosts.

After modfying LSB_MAX_PROBE_SBD, use `badmin mbdrestart` to restart mbatchd and let the modified value take effect.

If LSB_MAX_PROBE_SBD is defined, the value of MAX_SBD_FAILED in lsb.params can be less than 3.

Valid Values Any positive integer between 0 and 64

Default 20

See also MAX_SBD_FAIL in "lsb.params"

LSB_MAX_NQS_QUEUES

Syntax **LSB_MAX_NQS_QUEUES=***nqs_queues*

Description The maximum number of NQS queues allowed in the LSF cluster. Required for LSF to work with NQS. You must restart mbatchd if you change the value of LSB_MAX_NQS_QUEUES.

The total number of NQS queues configured by NQS_QUEUES in lsb.queues cannot exceed the value of LSB_MAX_NQS_QUEUES. NQS queues in excess of the maximum queues are ignored.

If you do not define LSB_MAX_NQS_QUEUES or define an incorrect value, LSF-NQS interoperation is disabled.

Valid Values Any positive integer

Default None

LSB_MBD_PORT

See "LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT" on page 555.

LSB_MC_CHKPNT_RERUN

Syntax **LSB_MC_CHKPNT_RERUN=y | n**

Description For checkpointable MultiCluster jobs, if a restart attempt fails, the job will be rerun from the beginning (instead of from the last checkpoint) without administrator or user intervention.

The submission cluster does not need to forward the job again. The execution cluster reports the job's new pending status back to the submission cluster, and the job is dispatched to the same host to restart from the beginning

Default n

LSB_MC_INITFAIL_MAIL

Syntax **LSB_MC_INITFAIL_MAIL=y | n**

Description MultiCluster job forwarding model only. Specify y to make LSF email the job owner when a job is suspended after reaching the retry threshold.

Default n

LSB_MC_INITFAIL_RETRY

Syntax **LSB_MC_INITFAIL_RETRY=***integer*

Description MultiCluster job forwarding model only. Defines the retry threshold and causes LSF to suspend a job that repeatedly fails to start. For example, specify 2 retry attempts to make LSF attempt to start a job 3 times before suspending it.

Default 5

LSB_MEMLIMIT_ENFORCE

Syntax **LSB_MEMLIMIT_ENFORCE=y | n**

Description Specify y to enable LSF memory limit enforcement.

If enabled, LSF sends a signal to kill all processes that exceed queue-level memory limits set by MEMLIMIT in lsb.queues or job-level memory limits specified by bsub -M *mem_limit*.

Otherwise, LSF passes memory limit enforcement to the OS. UNIX operating systems that support RLIMIT_RSS for setrlimit() can apply the memory limit to each process.

The following operating systems do not support memory limit at the OS level:

- Windows
- Sun Solaris 2.x

Default Undefined. LSF passes memory limit enforcement to the OS.

See also lsb.queues(5)

LSB_MIG2PEND

Syntax **LSB_MIG2PEND=0 | 1**

Description Applies only to migrating jobs.

If 1, requeues migrating jobs instead of restarting or rerunning them on the next available host. Requeues the jobs in the PEND state, in order of the original submission time, unless LSB_REQUEUE_TO_BOTTOM is also defined.

If you do not want migrating jobs to be run or restarted immediately, set LSB_MIG2PEND so that migrating jobs are considered as pending jobs and inserted in the pending jobs queue.

If you want migrating jobs to be considered as pending jobs but you want them to be placed at the bottom of the queue without considering submission time, define both LSB_MIG2PEND and LSB_REQUEUE_TO_BOTTOM.

Also considers job priority when requeuing jobs.

Does not work with MultiCluster.

Default Undefined

See also LSB_REQUEUE_TO_BOTTOM

LSB_MOD_ALL_JOBS

Syntax **LSB_MOD_ALL_JOBS=y** | **Y**

Description If set, enables `bmod` to modify resource limits and location of job output files for running jobs.

After a job has been dispatched, the following modifications can be made:

- CPU limit (`-c` [*hour*:]*minute*[/*host_name* | /*host_model*] | `-cn`)
- Memory limit (`-M` *mem_limit* | `-Mn`)
- Run limit (`-W` *run_limit*[/*host_name* | /*host_model*] | `-Wn`)
- Standard output file name (`-o` *output_file* | `-on`)
- Standard error file name (`-e` *error_file* | `-en`)
- Rerunnable jobs (`-r` | `-rn`)
- Termination time (`-t` | `-tn`

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 `lsf.conf`.

Default Undefined

See also LSB_JOB_CPULIMIT, LSB_JOB_MEMLIMIT

LSB_NCPU_ENFORCE

Description When set to 1, enables parallel fairshare (considers the number of CPUs when calculating dynamic priority).

Default Undefined

LSB_NQS_PORT

Syntax **LSB_NQS_PORT=***port_number*

Description Required for LSF to work with NQS.

TCP service port to use for communication with NQS.

Where defined This parameter can alternatively be set as an environment variable or in the services database such as `/etc/services`.

Example `LSB_NQS_PORT=607`

Default Undefined

LSB_PSET_BIND_DEFAULT

Syntax **LSB_PSET_BIND_DEFAULT=y | Y**

Description If set, Platform LSF HPC binds a job that is not explicitly associated with an HP-UX pset to the default pset 0. If LSB_PSET_BIND_DEFAULT is not set, LSF HPC must still attach the job to a pset, and so binds the job to the same pset used by the LSF HPC daemons.

Use LSB_PSET_BIND_DEFAULT to improve LSF daemon performance by automatically unbinding a job with no pset options from the pset used by the LSF daemons, and binding it to the default pset.

Default Undefined

LSB_QUERY_PORT

Syntax **LSB_QUERY_PORT=***port_number*

Description Optional. Applies only to UNIX platforms that support thread programming.

This parameter is recommended for busy clusters with many jobs and frequent query requests to increase `mbatchd` performance when you use the `bjobs` command.

This may indirectly increase overall `mbatchd` performance.

The `port_number` is the TCP/IP port number to be used by `mbatchd` to only service query requests from the LSF system. `mbatchd` checks the query port during initialization.

If LSB_QUERY_PORT *is not* defined:

- `mbatchd` uses the port specified by LSB_MBD_PORT in `lsf.conf`, or, if LSB_MBD_PORT is not defined, looks into the system services database for port numbers to communicate with other hosts in the cluster.
- For each query request it receives, `mbatchd` forks one child `mbatchd` to service the request. Each child `mbatchd` processes one request and then exits.

If LSB_QUERY_PORT is defined:

`mbatchd` prepares this port for connection.The default behavior of `mbatchd` changes, a child `mbatchd` is forked, and the child `mbatchd` creates threads to process requests.

`mbatchd` responds to requests by forking one child `mbatchd`. As soon as `mbatchd` has forked a child `mbatchd`, the child `mbatchd` takes over and listens on the port to process more query requests. For each request, the child `mbatchd` creates a thread to process it.

The interval used by `mbatchd` for forking new child `mbatchd`s is specified by the parameter MBD_REFRESH_TIME in `lsb.params`.

The child `mbatchd` continues to listen to the port number specified by LSB_QUERY_PORT and creates threads to service requests until the job changes status, a new job is submitted, or the time specified in MBD_REFRESH_TIME in lsb.params has passed (see "MBD_REFRESH_TIME" on page 388 for more details). When any of these happens, the parent mbatchd sends a message to the child mbatchd to exit.

Operating system support

See the Online Support area of the Platform Computing Web site at www.platform.com for the latest information about operating systems that support multithreaded mbatchd.

Default Undefined

See also MBD_REFRESH_TIME in "lsb.params".

LSB_REQUEUE_TO_BOTTOM

Syntax **LSB_REQUEUE_TO_BOTTOM=0 | 1**

Description Optional. If 1, requeues automatically requeued jobs to the bottom of the queue instead of to the top. Also requeues migrating jobs to the bottom of the queue if LSB_MIG2PEND is defined.

Does not work with MultiCluster.

Default Undefined

See also LSB_MIG2PEND, REQUEUE_EXIT_VALUES in "lsb.queues"

LSB_RLA_HOST_LIST

Syntax **LSB_RLA_HOST_LIST="***host_name ...***"**

Description By default, the LSF scheduler can contact the LSF HPC topology adapter (RLA) running on any host for Linux/QsNet RMS allocation requests. LSB_RLA_HOST_LIST defines a list of hosts to restrict which RLAs the LSF scheduler contacts.

If LSB_RLA_HOST_LIST is configured, you must list at least one host per RMS partition for the RMS partition to be considered for job scheduling.

Listed hosts must be defined in lsf.cluster.*cluster_name*.

Host names are separated by spaces.

Default Undefined.

LSB_RLA_PORT

Syntax **LSB_RLA_PORT=***port_number*

Description TCP port used for communication between the LSF HPC topology adapter (RLA) and the LSF HPC scheduler plugin.

Default 6883

LSB_RLA_UPDATE

Syntax **LSB_RLA_UPDATE=***seconds*

Description Specifies how often the LSF HPC scheduler refreshes free node information from the LSF HPC topology adapter (RLA).

Default 600 seconds

LSB_RLA_WORKDIR

Syntax **LSB_RLA_WORKDIR=***directory*

Description Directory to store the LSF HPC topology adapter (RLA) status file. Allows RLA to recover its original state when it restarts. When RLA first starts, it creates the directory defined by LSB_RLA_WORKDIR if it does not exist, then creates subdirectories for each host.

You should avoid using `/tmp` or any other directory that is automatically cleaned up by the system. Unless your installation has restrictions on the LSB_SHAREDIR directory, you should use the default for LSB_RLA_WORKDIR.

Default `LSB_SHAREDIR/`*`cluster_name`*`/rla_workdir`

LSB_RMSACCT_DELAY

Syntax **LSB_RMSACCT_DELAY=***time_seconds*

Description If set, RES waits the specified number of seconds before exiting to allow LSF and RMS job statistics to synchronize.

If LSB_RMSACCT_DELAY=0, RES waits forever until the database is up to date.

Default Undefined, RES does not wait at all.

LSB_RMS_MAXNUMNODES

Syntax **LSB_RMS_MAXNUMNODES=***integer*

Description Maximum number of nodes in a system. Specifies a maximum value for the `nodes` argument to the topology scheduler options specified in:

- `-extsched` option of `bsub`
- DEFAULT_EXTSCHED and MANDATORY_EXTSCHED in `lsb.queues`

Default 1024

LSB_RMS_MAXNUMRAILS

Syntax **LSB_RMS_MAXNUMRAILS=***integer*

Description Maximum number of rails in a system. Specifies a maximum value for the `rails` argument to the topology scheduler options specified in:

- `-extsched` option of `bsub`
- DEFAULT_EXTSCHED and MANDATORY_EXTSCHED in `lsb.queues`

Default 32

LSB_RMS_MAXPTILE

Syntax **LSB_RMS_MAXPTILE=***integer*

Description Maximum number of CPUs per node in a system. Specifies a maximum value for the `ptile` argument to the topology scheduler options specified in:

- `-extsched` option of `bsub`
- DEFAULT_EXTSCHED and MANDATORY_EXTSCHED in `lsb.queues`

Default 32

LSB_SLURM_BESTFIT

Syntax **LSB_SLURM_BESTFIT=y | Y**

Description Enables best-fit node allocation for HP XC SLURM jobs.

By default, LSF applies a *first-fit* allocation policy to select from the nodes available for the job. The allocations are made left to right for all parallel jobs, and right to left for all serial jobs (all other job requirements being equal).

In a heterogeneous XC machine, a *best-fit* allocation may be preferable for clusters where a mix of serial and parallel jobs run. In this context, best fit means: "the nodes that minimally satisfy the requirements." Nodes with the maximum number of CPUs are chosen first. For parallel and serial jobs, the nodes with minimal memory, minimal tmp space, and minimal weight are chosen.

Default Undefined

LSB_SBD_PORT

See "LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT" on page 555.

LSB_SET_TMPDIR

Syntax **LSB_SET_TMPDIR=y | n**

If `y`, LSF sets the TMPDIR environment variable, overwriting the current value with */tmp/job_ID*.

Default `n`

LSB_SHAREDIR

Syntax **LSB_SHAREDIR=***dir*

Description Directory in which the job history and accounting logs are kept for each cluster. These files are necessary for correct operation of the system. Like the organization under LSB_CONFDIR, there is one subdirectory for each cluster.

The LSB_SHAREDIR directory must be owned by the LSF administrator. It must be accessible from all hosts that can potentially become the master host, and must allow read and write access from the master host.

The LSB_SHAREDIR directory typically resides on a reliable file server.

Default `LSF_INDEP/work`

See also LSB_LOCALDIR

LSB_SHORT_HOSTLIST

Syntax **LSB_SHORT_HOSTLIST=1**

Description Displays an abbreviated list of hosts in `bjobs` and `bhist` for a parallel job where multiple processes of a job are running on a host. Multiple processes are displayed in the following format:

`processes``*hostA`

For example, if a parallel job is running 5 processes on `hostA`, the information is displayed in the following manner:

`5*hostA`

Setting this parameter may improve `mbatchd` restart performance and accelerate event replay.

Default Undefined

LSB_SIGSTOP

Syntax **`LSB_SIGSTOP=`***signal_name* | *signal_value*

Description Specifies the signal sent by the SUSPEND action in LSF. You can specify a signal name or a number.

If LSB_SIGSTOP is set to anything other than SIGSTOP, the SIGTSTP signal that is normally sent by the SUSPEND action is not sent.

If this parameter is undefined, by default the SUSPEND action in LSF sends the following signals to a job:

- Parallel or interactive jobs—1. SIGTSTP is sent first to allow user programs to catch the signal and clean up. 2. SIGSTOP is sent 10 seconds after SIGTSTP. SIGSTOP cannot be caught by user programs.
- Other jobs—SIGSTOP is sent. SIGSTOP cannot be caught by user programs.

The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the `kill -l` command.

Example `LSB_SIGSTOP=SIGKILL`

In this example, the SUSPEND action sends the three default signals sent by the TERMINATE action (SIGINT, SIGTERM, and SIGKILL) 10 seconds apart.

Default Undefined. Default SUSPEND action in LSF is sent.

LSB_SUB_COMMANDNAME

Syntax **`LSB_SUB_COMMANDNAME=y`** | **`Y`**

Description If set, enables `esub` to use the variable LSB_SUB_COMMAND_LINE in the `esub` job parameter file specified by the $LSB_SUB_PARM_FILE environment variable.

The LSB_SUB_COMMAND_LINE variable carries the value of the `bsub` command argument, and is used when `esub` runs.

Example `esub` contains:

```
#!/bin/sh
. $LSB_SUB_PARM_FILE
exec 1>&2
if [ $LSB_SUB_COMMAND_LINE="netscape" ]; then
echo "netscape is not allowed to run in batch mode"
exit $LSB_SUB_ABORT_VALUE
fi
```

LSB_SUB_COMMAND_LINE is defined in $LSB_SUB_PARM_FILE as:

`LSB_SUB_COMMAND_LINE=netscape`

A job submitted with:

`bsub netscape ...`

Causes esub to echo the message:

`netscape is not allowed to run in batch mode`

Default Undefined

See also LSB_SUB_COMMAND_LINE and LSB_SUB_PARM_FILE environment variables

LSB_STDOUT_DIRECT

Syntax **LSB_STDOUT_DIRECT=y** | **Y**

Description When set, and used with the `-o` or `-e` options of `bsub`, redirects standard output or standard error from the job directly to a file as the job runs.

If LSB_STDOUT_DIRECT is not set and you use the `bsub -o` option, the standard output of a job is written to a temporary file and copied to the file you specify *after* the job finishes.

LSB_STDOUT_DIRECT is not supported on Windows.

Default Undefined

LSB_TIME_CMD

Syntax **LSB_TIME_CMD=***timimg_level*

Description The timing level for checking how long batch commands run.

Time usage is logged in milliseconds; specify a positive integer.

Example: `LSB_TIME_CMD=1`

Default Undefined

See also LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_MBD

Syntax **LSB_TIME_MBD=***timing_level*

Description The timing level for checking how long `mbatchd` routines run.

Time usage is logged in milliseconds; specify a positive integer.

Example: `LSB_TIME_MBD=1`

Default Undefined

See also LSB_TIME_CMD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_RESERVE_NUMJOBS

Syntax **LSB_TIME_RESERVE_NUMJOBS=***maximum_reservation_jobs*

Description Enables time-based slot reservation. The value must be positive integer.

LSB_TIME_RESERVE_NUMJOBS controls maximum number of jobs using time-based slot reservation. For example, if LSB_TIME_RESERVE_NUMJOBS=4, only the top 4 jobs will get their future allocation information.

Use LSB_TIME_RESERVE_NUMJOBS=1 to allow only the highest priority job to get accurate start time prediction.

Recommended value 3 or 4 is the recommended setting. Larger values are not as useful because after the first pending job starts, the estimated start time of remaining jobs may be changed.

Default Undefined

LSB_TIME_SBD

Syntax **LSB_TIME_SBD=***timing_level*

Description The timing level for checking how long `sbatchd` routines run.

Time usage is logged in milliseconds; specify a positive integer.

Example: `LSB_TIME_SBD=1`

Default Undefined

See also LSB_TIME_CMD, LSB_TIME_MBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_SCH

Syntax **LSB_TIME_SCH=***timing_level*

Description The timing level for checking how long `mbschd` routines run.

Time usage is logged in milliseconds; specify a positive integer.

Example: `LSB_TIME_SCH=1`

Default Undefined

LSB_UTMP

Syntax **LSB_UTMP=y | Y**

Description If set, enables registration of user and account information for interactive batch jobs submitted with `bsub -Ip` or `bsub -Is`. To disable `utmp` file registration, set LSB_UTMP to any value other than `y` or `Y`; for example, `LSB_UTMP=N`.

LSF registers interactive batch jobs the job by adding a entries to the `utmp` file on the execution host when the job starts. After the job finishes, LSF removes the entries for the job from the `utmp` file.

Limitations Registration of `utmp` file entries is supported only on SGI IRIX (6.4 and later).

`utmp` file registration is not supported in a MultiCluster environment.

Because interactive batch jobs submitted with `bsub -I` are not associated with a pseudo-terminal, `utmp` file registration is not supported for these jobs.

Default Undefined

LSF_AFS_CELLNAME

Syntax **LSF_AFS_CELLNAME=***AFS_cell_name*

Description Must be defined to AFS cell name if the AFS file system is in use.

Example:

`LSF_AFS_CELLNAME=cern.ch`

Default Undefined

LSF_AM_OPTIONS

Syntax **LSF_AM_OPTIONS=AMFIRST | AMNEVER**

Description Determines the order of file path resolution when setting the user's home directory.

This variable is rarely used but sometimes LSF does not properly change the directory to the user's home directory when the user's home directory is automounted. Setting LSF_AM_OPTIONS forces LSF to change directory to $HOME before attempting to automount the user's home.

When this parameter is undefined or set to AMFIRST, LSF:

- Sets the user's $HOME directory from the automount path. If it cannot do so, LSF sets the user's $HOME directory from the `passwd` file.

When this parameter is set to AMNEVER, LSF:

- Never uses automount to set the path to the user's home. LSF sets the user's $HOME directory directly from the `passwd` file.

Valid Values The two values are AMFIRST and AMNEVER

Default Undefined; same as AMFIRST

LSF_API_CONNTIMEOUT

Syntax **LSF_API_CONNTIMEOUT=***time_seconds*

Description Timeout when connecting to LIM.

Default 5

See also LSF_API_RECVTIMEOUT

LSF_API_RECVTIMEOUT

Syntax **LSF_API_RECVTIMEOUT=***time_seconds*

Description Timeout when receiving a reply from LIM.

Default 20

See also LSF_API_CONNTIMEOUT

LSF_AUTH

Syntax **LSF_AUTH=eauth | ident**

Description Optional. Determines the type of authentication used by LSF.

External user authentication is configured automatically during installation (LSF_AUTH=eauth). If LSF_AUTH is not defined, privileged ports (setuid) authentication is used. This is the mechanism most UNIX remote utilities use.

External authentication is the only way to provide security for clusters that contain Windows hosts.

If this parameter is changed, you must shut down and restart all LSF daemons by running `lsf_daemons start` on each LSF server host so that all daemons use the new authentication method.

When LSF uses privileged ports for user authentication, LSF commands must be installed as setuid programs owned by root to operate correctly. If the commands are installed in an NFS-mounted shared file system, the file system must be mounted with setuid execution allowed (that is, without the `nosuid` option). See the man page for the `mount` command for more details.

Windows does not have the concept of setuid binaries and does not restrict access to privileged ports, so the undefined method does not provide any security on Windows.

Valid values

- `eauth`

  For site-specific external authentication.

- `ident`

  For authentication using the RFC 931/1413/1414 protocol to verify the identity of the remote client.

  If LSF_AUTH is defined as `ident`, RES uses the RFC 1413 identification protocol to verify the identity of the remote user. RES is also compatible with the older RFC 931 authentication protocol. The name, `ident`, must be registered in the system services database.

setuid is not a valid value for LSF_AUTH. For privileged ports authentication, LSF_AUTH must not be defined at all in lsf.conf.

Default `eauth` (configured automatically during installation)

LSF_AUTH_DAEMONS

Syntax **LSF_AUTH_DAEMONS=***any_value*

Description Enables daemon authentication, as long as LSF_AUTH in `lsf.conf` is set to `eauth`. Daemons will call `eauth` to authenticate each other.

Default Undefined

LSF_BINDIR

Syntax **LSF_BINDIR=***dir*

Description Directory in which all LSF user commands are installed.

Default LSF_MACHDEP/bin

LSF_CMD_LOGDIR

Syntax **LSF_CMD_LOGDIR=***path*

Description The path to the log files used for debugging LSF commands.

This parameter can also be set from the command line.

Default /tmp

See also LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSF_CMD_LOG_MASK

Syntax **LSF_CMD_LOG_MASK=***log_level*

Description Specifies the logging level of error messages from LSF commands.

For example:

```
LSF_CMD_LOG_MASK=LOG_DEBUG
```

To specify the logging level of error messages, use LSB_CMD_LOG_MASK. To specify the logging level of error messages for LSF daemons, use LSF_LOG_MASK.

LSF commands log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSF_CMD_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

The commands log to the `syslog` facility unless LSF_CMD_LOGDIR is set.

Valid values The log levels from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Default LOG_WARNING

See also LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSF_CONF_RETRY_INT

Syntax **LSF_CONF_RETRY_INT=***time_seconds*

Description The number of seconds to wait between unsuccessful attempts at opening a configuration file (only valid for LIM). This allows LIM to tolerate temporary access failures.

Default 30

See also LSF_CONF_RETRY_MAX

LSF_CONF_RETRY_MAX

Syntax **LSF_CONF_RETRY_MAX=***integer*

Description The maximum number of unsuccessful attempts at opening a configuration file (only valid for LIM). This allows LIM to tolerate temporary access failures.

Default 0

See also LSF_CONF_RETRY_INT

LSF_CONFDIR

Syntax **LSF_CONFDIR=***dir*

Description Directory in which all LSF configuration files are installed. These files are shared throughout the system and should be readable from any host. This directory can contain configuration files for more than one cluster.

The files in the LSF_CONFDIR directory must be owned by the primary LSF administrator, and readable by all LSF server hosts.

Default `LSF_INDEP/conf`

See also LSB_CONFDIR

LSF_DAEMON_WRAP

Syntax **LSF_DAEMON_WRAP=y | Y**

Description Applies only to DCE/DFS and AFS environments; if you are installing LSF on a DCE or AFS environment, set this parameter to `y` or `Y`.

When this parameter is set to `y` or `Y`, `mbatchd`, `sbatchd`, and RES run the executable `daemons.wrap` in LSF_SERVERDIR.

Default Undefined

LSF_DEBUG_LIM

Syntax **LSF_DEBUG_LIM=***log_class*

Description Sets the log class for debugging LIM.

Specifies the log class filtering that will be applied to LIM. Only messages belonging to the specified log class are recorded.

The LSF_DEBUG_LIM sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSF_DEBUG_LIM=LC_TRACE
```

You need to restart the daemons after setting LSF_DEBUG_LIM for your changes to take effect.

If you use the command `lsadmin limdebug` to temporarily change this parameter without changing `lsf.conf`, you will not need to restart the daemons.

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSF_DEBUG_LIM="LC_TRACE LC_EXEC"
```

This parameter can also be defined from the command line.

Valid values

Valid log classes are:

- LC_AFS - Log AFS messages
- LC_AUTH - Log authentication messages
- LC_CHKPNT - log checkpointing messages
- LC_COMM - Log communication messages
- LC_DCE - Log messages pertaining to DCE support
- LC_EXEC - Log significant steps for job execution
- LC_FILE - Log file transfer messages
- LC_HANG - Mark where a program might hang
- LC_LICENCE - Log licence management messages
- LC_MULTI - Log messages pertaining to MultiCluster
- LC_PIM - Log PIM messages
- LC_SIGNAL - Log messages pertaining to signals
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

Default

Undefined

See also

LSF_DEBUG_RES, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_DEBUG_RES

Syntax

LSF_DEBUG_RES=*log_class*

Description

Sets the log class for debugging RES.

Specifies the log class filtering that will be applied to RES. Only messages belonging to the specified log class are recorded.

LSF_DEBUG_RES sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG
LSF_DEBUG_RES=LC_TRACE
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSF_DEBUG_RES="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSF_DEBUG_RES for your changes to take effect.

If you use the command `lsadmin resdebug` to temporarily change this parameter without changing `lsf.conf`, you will not need to restart the daemons.

Valid Values For a list of valid log classes see LSF_DEBUG_LIM

Default Undefined

See also LSF_DEBUG_LIM, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_DHCP_ENV

Syntax **`LSF_DHCP_ENV=y`**

Description If defined, enables dynamic IP addressing for all LSF client hosts in the cluster.

Dynamic IP addressing is not supported across clusters in a MultiCluster environment.

If you set LSF_DHCP_ENV, you must also specify LSF_DYNAMIC_HOST_WAIT_TIME in order for hosts to rejoin a cluster after their IP address changes.

After or chainging this parameter, you must run `lsadmin reconfig` and `badmin mbdrestart` to restart all LSF daemons.

Default Undefined

See also LSF_DYNAMIC_HOST_WAIT_TIME

LSF_DISPATCHER_LOGDIR

Syntax **`LSF_DISPATCHER_LOGDIR=`***path*

Description Specifies the path to the log files for slot allocation decsions for queue-based fairshare.

If defined, LSF writes the results of its queue-based fairshare slot calculation to the specified directory. Each line in the file consists of a timestamp for the slot allocation and the number of slots allocated to each queue under its control. LSF logs in this file every minute. The format of this file is suitable for plotting with `gnuplot`.

If you set LSF_DHCP_ENV, you must also specify LSF_DYNAMIC_HOST_WAIT_TIME in order for hosts to rejoin a cluster after their IP address changes.

Example

```
# clients managed by LSF
# Roma # Verona # Genova # Pisa # Venezia # Bologna
  15/3      19:4:50   0 0 0 0 0 0
  15/3      19:5:51   8 5 2 5 2 0
  15/3      19:6:51   8 5 2 5 5 1
  15/3      19:7:53   8 5 2 5 5 5
  15/3      19:8:54   8 5 2 5 5 0
  15/3      19:9:55   8 5 0 5 4 2
```

The queue names are in the header line of the file. The columns correspond to the allocations per each queue.

Default Not defined

LSF_DYNAMIC_HOST_WAIT_TIME

Syntax **LSF_DYNAMIC_HOST_WAIT_TIME=***time_seconds*

Description Defines the period of time from startup for dynamic slave LIMs (hosts) to wait for an acknowledgement from the master LIM. This signals to the dynamic host that it is already in the cluster and therefore does not need to be added. If it does not receive this acknowledgement, the dynamic host sends a request to the master LIM to add it to the cluster.

To enable dynamically added hosts, you must define both LSF_DYNAMIC_HOST_WAIT_TIME in lsf.conf, and LSF_HOST_ADDR_RANGE in lsf.cluster.*cluster_name*.

Recommended value Up to 60 seconds for every 1000 hosts in the cluster, for a maximum of 15 minutes. Selecting a smaller value will result in a quicker response time for new hosts at the expense of an increased load on the master LIM.

Example `LSF_DYNAMIC_HOST_WAIT_TIME=60`

Hosts will wait 60 seconds from startup to receive an acknowledgement from the master LIM. If it does not receive the acknowledgement within the 60 seconds, it will send a request for the master LIM to add it to the cluster.

Default INFINIT_INT (the host will never send a request to the master LIM)

LSF_ENABLE_CSA

Syntax **LSF_ENABLE_CSA=y | Y**

Description If set, enables LSF to write records for LSF jobs to IRIX 6.5.9 Comprehensive System Accounting facility (CSA).

The IRIX 6.5.9 Comprehensive System Accounting facility (CSA) writes an accounting record for each process in the `pacct` file, which is usually located in the `/var/adm/acct/day` directory. IRIX system administrators then use the `csabuild` command to organize and present the records on a job by job basis.

When LSF_ENABLE_CSA is set, for each job run on the IRIX system, LSF writes an LSF-specific accounting record to CSA when the job starts, and when the job finishes. LSF daemon accounting in CSA starts and stops with the LSF daemon.

To disable IRIX CSA accounting, remove LSF_ENABLE_CSA from `lsf.conf`.

See the IRIX 6.5.9 resource administration documentation for information about CSA.

Setting up IRIX CSA

1. Define the LSF_ENABLE_CSA parameter in `lsf.conf`:

```
...
LSF_ENABLE_CSA=Y
...
```

2. Set the following parameters in `/etc/csa.conf` to `on`:
   - CSA_START
   - WKMG_START
3. Run the `csaswitch` command to turn on the configuration changes in `/etc/csa.conf`.

See the IRIX 6.5.9 resource administration documentation for information about the `csaswitch` command.

Information written to the pacct file

LSF writes the following records to the `pacct` file when a job starts and when it exits:

- Job record type (job start or job exit)
- Current system clock time
- Service provider (LSF)
- Submission time of the job (at job start only)
- User ID of the job owner
- Array Session Handle (ASH) of the job
- IRIX job ID
- IRIX project ID
- LSF job name if it exists
- Submission host name
- LSF queue name
- LSF external job ID
- LSF job array index
- LSF job exit code (at job exit only)
- NCPUS—number of CPUs the LSF job has been using

Default Undefined

LSF_ENABLE_DUALCORE

Syntax **`LSF_ENABLE_DUALCORE=y | n`**

Description

Enables job scheduling based on dual-core CPU information for a host. If yes (Y), LSF scheduling policies use the detected number of CPU cores as the number of physical CPUs on the host instead of the number of physical CPUs for job scheduling. For a dual-core host, lshosts shows the number of cores under ncpus instead of physical CPUs.

To make use of dual-core CPUs for scheduling, hosts must have the `lsf_dualcore_x86` license feature enabled. Each dual core processor requires one standard LSF license and one `lsf_dualcore_x86` license. Use lshosts -l to see the number of dual-core licenses enabled and needed.

Default N

LSF_ENABLE_EXTSCHEDULER

Syntax **LSF_ENABLE_EXTSCHEDULER=y | Y**

Description If set, enables `mbatchd` external scheduling for LSF HPC.

Default Undefined

LSF_ENVDIR

Syntax **LSF_ENVDIR=***dir*

Description Directory containing the `lsf.conf` file.

By default, `lsf.conf` is installed by creating a shared copy in LSF_CONFDIR and adding a symbolic link from `/etc/lsf.conf` to the shared copy. If LSF_ENVDIR is set, the symbolic link is installed in `LSF_ENVDIR/lsf.conf`.

The `lsf.conf` file is a global environment configuration file for all LSF services and applications. The LSF default installation places the file in LSF_CONFDIR.

Default `/etc`

LSF_EVENT_PROGRAM

Syntax **LSF_EVENT_PROGRAM=***event_program_name*

Description Specifies the name of the LSF event program to use.

If a full path name is not provided, the default location of this program is LSF_SERVERDIR.

If a program that does not exist is specified, event generation will not work.

If this parameter is undefined, the default name is `genevent` on UNIX

If this parameter is undefined, the default name is `genevent.exe` on Windows.

Default Undefined

LSF_EVENT_RECEIVER

Syntax **LSF_EVENT_RECEIVER=***event_receiver_program_name*

Description Specifies the LSF event receiver and enables event generation.

Any string may be used as the LSF event receiver; this information is not used by LSF to enable the feature but is only passed as an argument to the event program.

If LSF_EVENT_PROGRAM specifies a program that does not exist, event generation will not work.

If this parameter is undefined, event generation is disabled.

Default Undefined

LSF_HPC_EXTENSIONS

Syntax **LSF_HPC_EXTENSIONS="***extension_name ...***"**

Description Enables Platform LSF HPC extensions.

Valid values The following extension names are supported:

- CUMULATIVE_RUSAGE—when a parallel job script runs multiple `pam` commands, resource usage is collected for jobs in the job script, rather than being overwritten when each `pam` command is executed.
- DISP_RES_USAGE_LIMITS—`bjobs` displays resource usage limits configured in the queue as well as job-level limits.
- LSB_HCLOSE_BY_RES—`If res` is down, host is closed with a message

  `Host is closed because RES is not available.`

  The status of the closed host is `closed_Adm`. No new jobs are dispatched to this host, but currently running jobs are not suspended.
- RESERVE_BY_STARTTIME—LSF selects the reservation that will give the job the earliest predicted start time.

  By default, if multiple host groups are available for reservation, LSF chooses the largest possible reservation based on number of slots.
- SHORT_EVENTFILE—compresses long host name lists when event records are written to `lsb.events` and `lsb.acct` for large parallel jobs. The short host string has the format:

  `number_of_hosts*real_host_name`

  When SHORT_EVENTFILE is enabled, older daemons and commands (pre-LSF Version 6.2) cannot recognize the lsb.acct and lsb.events file format.

  For example, if the original host list record is

  `6 "hostA" "hostA" "hostA" "hostA" "hostB" "hostC"`

  redundant host names are removed and the short host list record becomes

  `3 "4*hostA" "hostB" "hostC"`

  When LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is set, and LSF reads the host list from `lsb.events` or `lsb.acct`, the compressed host list is expanded into a normal host list.

  SHORT_EVENTFILE affects the following events and fields:

  - JOB_START in `lsb.events` when a normal job is dispatched
    - `numExHosts` (%d)
    - `execHosts` (%s)
  - JOB_CHUNK in `lsb.events` when a job is inserted into a job chunk
    - `numExHosts` (%d)
    - `execHosts` (%s)
  - JOB_FORWARD in `lsb.events` when a job is forwarded to a MultiCluster leased host
    - `numReserHosts` (%d)
    - `reserHosts` (%s)
  - JOB_FINISH record in `lsb.acct`
    - `numExHosts` (%d)
    - `execHosts` (%s)

- SHORT_PIDLIST—shortens the output from `bjobs` to omit all but the first process ID (PID) for a job. `bjobs` displays only the first ID and a count of the process group IDs (PGIDs) and process IDs for the job.

  Without SHORT_PIDLIST, `bjobs -l` displays all the PGIDs and PIDs for the job. With SHORT_PIDLIST set, `bjobs -l` displays a count of the PGIDS and PIDs.
- TASK_MEMLIMIT—enables enforcment of a memory limit (`bsub -M`, `bmod -M`, or MEMLIMIT in `lsb.queues`) for individual tasks in a parallel job. If any parallel task exceeds the memory limit, LSF terminates the entire job.
- TASK_SWAPLIMIT—enables enforcment of a virtual memory (swap) limit (`bsub -v`, `bmod -v`, or SWAPLIMIT in `lsb.queues`) for individual tasks in a parallel job. If any parallel task exceeds the swap limit, LSF terminates the entire job.

Example JOB_START events in lsb.events:

For a job submitted with

```
% bsub -n 64 -R "span[ptile=32]" sleep 100
```

- *Without* SHORT_EVENTFILE, a JOB_START event like the following would be logged in `lsb.events`:

```
"JOB_START" "6.2" 1058989891 710 4 0 0 10.3 64 "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "u050" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"" "" 0 "" 0
```

- *With* SHORT_EVENTFILE, a JOB_START event would be logged in `lsb.events` with the number of execution hosts (`numExHosts` field) changed from 64 to 2 and the execution host list (`execHosts` field) shortened to `"32*hostA"` and `"32*hostB"`:

```
"JOB_START" "6.2" 1058998174 812 4 0 0 10.3 2 "32*hostA" "32*hostB" "" "" 0 ""
0 ""
```

Example JOB_FINISH records in lsb.acct:

For a job submitted with

```
% bsub -n 64 -R "span[ptile=32]" sleep 100
```

- *Without* SHORT_EVENTFILE, a JOB_FINISH event like the following would be logged in `lsb.acct`:

```
"JOB_FINISH" "6.2" 1058990001 710 33054 33816578 64 1058989880 0 0 1058989891
"user1" "normal" "span[ptile=32]" "" "" "hostA" "/scratch/user1/work" "" "" ""
"1058989880.710" 0 64 "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" 64 10.3 "" "sleep 100"
0.079999 0.270000 0 0 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 -1 "" "default" 0 64 "" ""
0 4304 6024 "" "" ""
```

- *With* SHORT_EVENTFILE, a JOB_FINISH event like the following would be logged in `lsb.acct` with the number of execution hosts (`numExHosts` field) changed from 64 to 2 and the execution host list (`execHosts` field) shortened to `"32*hostA"` and `"32*hostB"`:

```
"JOB_FINISH" "6.0" 1058998282 812 33054 33816578 64 1058998163 0 0 1058998174
"user1" "normal" "span[ptile=32]" "" "" "hostA" "/scratch/user1/work" "" "" ""
"1058998163.812" 0 2 "32*hostA" "32*hostB" 64 10.3 "" "sleep 100" 0.039999
0.259999 0 0 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 -1 "" "default" 0 64 "" "" 0 4304
6024 "" "" "" "" 0
```

Example bjobs -l ouput without SHORT_PIDLIST:

`bjobs -l` displays all the PGIDs and PIDs for the job:

```
% bjobs -l
Job <109>, User <user3>, Project <default>, Status <RUN>, Queue <normal>, Inte
                     ractive mode, Command <./myjob.sh>
Mon Jul 21 20:54:44: Submitted from host <hostA>, CWD <$HOME/LSF/jobs;

 RUNLIMIT
 10.0 min of hostA

 STACKLIMIT CORELIMIT MEMLIMIT
   5256 K    10000 K    5000 K
Mon Jul 21 20:54:51: Started on <hostA>;
Mon Jul 21 20:55:03: Resource usage collected.
                     MEM: 2 Mbytes;  SWAP: 15 Mbytes
                     PGID: 256871;  PIDs: 256871
                     PGID: 257325;  PIDs: 257325 257500 257482 257501 257523
                     257525 257531


 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -


<< Job <109> is done successfully. >>
```

Example bjobs -l ouput with SHORT_PIDLIST:

`bjobs -l` displays a count of the PGIDS and PIDs:

```
% bjobs -l
Job <109>, User <user3>, Project <default>, Status <RUN>, Queue <normal>, Inte
                     ractive mode, Command <./myjob.sh>
Mon Jul 21 20:54:44: Submitted from host <hostA>, CWD <$HOME/LSF/jobs;

 RUNLIMIT
 10.0 min of hostA

 STACKLIMIT CORELIMIT MEMLIMIT
   5256 K    10000 K    5000 K
Mon Jul 21 20:54:51: Started on <hostA>;
Mon Jul 21 20:55:03: Resource usage collected.
                     MEM: 2 Mbytes;  SWAP: 15 Mbytes
                     PGID(s):  256871:1 PID, 257325:7 PIDs

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -
```

Default Undefined

LSF_HPC_NCPU_COND

Syntax **LSF_HPC_NCPU_COND=and | or**

Description Defines how any two LSF_HPC_NCPU_* thresholds are combined.

Default `or`

LSF_HPC_NCPU_INCREMENT

Syntax **LSF_HPC_NCPU_INCREMENT=***increment*

Description Defines the upper limit for the number of CPUs that are changed since the last checking cycle.

Default 0

LSF_HPC_NCPU_INCR_CYCLES

Syntax **LSF_HPC_NCPU_INCR_CYCLES=***incr_cyscles*

Description Minimum number of consecutive cycles where the number of CPUs changed does not exceed LSF_HPC_NCPU_INCREMENT. LSF checks total usable CPUs every 2 minutes.

Default 1

LSF_HPC_NCPU_THRESHOLD

Syntax **LSF_HPC_NCPU_THRESHOLD=***threshold*

Description LSF_HPC_NCPU_THRESHOLD=threshold

The percentage of total usable CPUs in the LSF partition of an HP XC system.

Default 80

LSF_HPC_PJL_LOADENV_TIMEOUT

Syntax **LSF_HPC_PJL_LOADENV_TIMEOUT=***seconds*

Description Timeout value in seconds for PJL to load or unload the environment. For example, set LSF_HPC_PJL_LOADENV_TIMEOUT to the number of seconds needed for IBM POE to load or unload adapter windows.

At job startup, the PJL times out if the first task fails to register with PAM within the specified timeout value. At job shutdown, the PJL times out if it fails to exit after the last Taskstarter termination report within the specified timeout value.

Default LSF_HPC_PJL_LOADENV_TIMEOUT=300

LSF_ID_PORT

Syntax **LSF_ID_PORT=***port_number*

Description The network port number used to communicate with the authentication daemon when LSF_AUTH is set to `ident`.

LSF_INCLUDEDIR

Syntax **LSF_INCLUDEDIR=***dir*

Description Directory under which the LSF API header files `lsf.h` and `lsbatch.h` are installed.

Default `LSF_INDEP/include`

See also LSF_INDEP

LSF_INDEP

Syntax **LSF_INDEP=***dir*

Description Specifies the default top-level directory for all machine-independent LSF files.

This includes man pages, configuration files, working directories, and examples. For example, defining LSF_INDEP as `/usr/share/lsf/mnt` places man pages in `/usr/share/lsf/mnt/man`, configuration files in `/usr/share/lsf/mnt/conf`, and so on.

The files in LSF_INDEP can be shared by all machines in the cluster.

As shown in the following list, LSF_INDEP is incorporated into other LSF environment variables.

- LSB_SHAREDIR=$LSF_INDEP/work
- LSF_CONFDIR=$LSF_INDEP/conf
- LSF_INCLUDEDIR=$LSF_INDEP/include
- LSF_MANDIR=$LSF_INDEP/man
- XLSF_APPDIR=$LSF_INDEP/misc

Default `/usr/share/lsf/mnt`

See also LSF_MACHDEP, LSB_SHAREDIR, LSF_CONFDIR, LSF_INCLUDEDIR, LSF_MANDIR, XLSF_APPDIR

LSF_INTERACTIVE_STDERR

Syntax **LSF_INTERACTIVE_STDERR=y | n**

Description Separates `stderr` from `stdout` for interactive tasks and interactive batch jobs.

This is useful to redirect output to a file with regular operators instead of the `bsub -e err_file` and `-o` *`out_file`* options.

This parameter can also be enabled or disabled as an environment variable.

WARNING **If you enable this parameter globally in lsf.conf, check any custom scripts that manipulate `stderr` and `stdout`.**

When this parameter is undefined or set to `n`, the following are written to `stdout` on the submission host for interactive tasks and interactive batch jobs:

- Job standard output messages
- Job standard error messages

The following are written to `stderr` on the submission host for interactive tasks and interactive batch jobs:

- LSF messages
- NIOS standard messages
- NIOS debug messages (if LSF_NIOS_DEBUG=1 in `lsf.conf`)

When this parameter is set to `y`, the following are written to `stdout` on the submission host for interactive tasks and interactive batch jobs:

- Job standard output messages
- The following are written to `stderr` on the submission host:
- Job standard error messages
- LSF messages
- NIOS standard messages
- NIOS debug messages (if LSF_NIOS_DEBUG=1 in `lsf.conf`)

Default Undefined

Notes When this parameter is set, the change affects interactive tasks and interactive batch jobs run with the following commands:

- `bsub -I`
- `bsub -Ip`
- `bsub -Is`
- `lsrun`
- `lsgrun`
- `lsmake` (Platform Make)
- `bsub pam` (Platform LSF HPC)

Limitations

- Pseudo-terminal—Do not use this parameter if your application depends on `stderr` as a terminal. This is because LSF must use a non-pseudo-terminal connection to separate `stderr` from `stdout`.
- Synchronization—Do not use this parameter if you depend on messages in `stderr` and `stdout` to be synchronized and jobs in your environment are continuously submitted. A continuous stream of messages causes `stderr` and `stdout` to not be

synchronized. This can be emphasized with parallel jobs. This situation is similar to that of `rsh`.

- NIOS standard and debug messages—NIOS standard messages, and debug messages (when LSF_NIOS_DEBUG=1 in `lsf.conf` or as an environment variable) are written to `stderr`. NIOS standard messages are in the format `<<message>>`, which makes it easier to remove them if you wish. To redirect NIOS debug messages to a file, define LSF_CMD_LOGDIR in `lsf.conf` or as an environment variable.

See also LSF_NIOS_DEBUG, LSF_CMD_LOGDIR

LSF_IRIX_BESTCPUS (OBSOLETE)

LSF_IRIX_BESTCPUS is obsolete. Use LSB_CPUSET_BESTCPUS.

LSF_LD_SECURITY

Syntax **`LSF_LD_SECURITY=y | n`**

Description When you activate this parameter, jobs submitted using `bsub -Is` have the environment variables LD_PRELOAD and LD_LIBRARY_PATH removed from the user's job environment to ensure enhanced security against users obtaining root privileges. `bsub -Is` submits an interactive job and creates a pseudo-terminal with shell mode support when the job starts.

Default `N`

LSF_LIBDIR

Syntax **`LSF_LIBDIR=`***dir*

Description Specifies the directory in which the LSF libraries are installed. Library files are shared by all hosts of the same type.

Default `LSF_MACHDEP/lib`

LSF_LIC_SCHED_HOSTS

Syntax **`LSF_LIC_SCHED_HOSTS="`***candidate_host_list***`"`**

`candidate_host_list` is a space-separated list of hosts that are candidate LSF License Scheduler hosts.

Description The candidate License Scheduler host list is read by LIM on each host to check if the host is a candidate License Scheduler master host. If the host is on the list, LIM starts the License Scheduler daemon (`bld`) on the host.

LSF_LIC_SCHED_PREEMPT_REQUEUE

Syntax **`LSF_LIC_SCHED_PREEMPT_REQUEUE=y | n`**

Description Set this parameter to requeue a job whose license is preempted by LSF License Scheduler. The job will be killed and requeued instead of suspended.

If you set `LSF_LIC_SCHED_PREEMPT_REQUEUE`, do not set `LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE`. If both these parameters are set, `LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE` is ignored.

Default N

See Also LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, LSF_LIC_SCHED_PREEMPT_STOP

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE

Syntax **LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE=y | n**

Description Set this parameter to release the slot of a job that is suspended when the its license is preempted by LSF License Scheduler.

If you set `LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE`, do not set `LSF_LIC_SCHED_PREEMPT_REQUEUE`. If both these parameters are set, `LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE` is ignored.

Default Y

See Also LSF_LIC_SCHED_PREEMPT_REQUEUE, LSF_LIC_SCHED_PREEMPT_STOP

LSF_LIC_SCHED_PREEMPT_STOP

Syntax **LSF_LIC_SCHED_PREEMPT_STOP=y | n**

Description Set this parameter to use job controls to stop a job that is preempted. When this parameter is set, a UNIX SIGSTOP signal is sent to suspend a job instead of a UNIX SIGTSTP.

To send a SIGSTOP signal instead of SIGTSTP, the following parameter in `lsb.queues` must also be set:

`JOB_CONTROLS=SUSPEND[`**`SIGSTOP`**`]`

Default N

See Also LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, LSF_LIC_SCHED_PREEMPT_REQUEUE

LSF_LICENSE_ACCT_PATH

Syntax **LSF_LICENSE_ACCT_PATH=***dir*

Description Specifies the location for the license accounting files. These include the license accounting files for LSF Family products.

Use this parameter to define the location of all the license accounting files. By defining this parameter, you can store the license accounting files for the LSF Family of products in the same directory for convenience.

Default Undefined. The license accounting files are stored in the default log directory for the particular product. For example, LSF stores its license audit file in the LSF system log file directory, while LSF License Scheduler stores its license audit file in the `LSF_SHAREDIR/db` directory.

See also

- LSF_LOGDIR
- `lsf.cluster_name.license.acct`
- `bld.license.acct`

LSF_LICENSE_FILE

Syntax **LSF_LICENSE_FILE="***file_name ... | port_number@host_name***"**

Description Specifies one or more demo or FLEXnet-based permanent license files used by LSF.

The value for LSF_LICENSE_FILE can be either of the following:

- The full path name to the license file.

  UNIX example:

  `LSF_LICENSE_FILE=/usr/share/lsf/cluster1/conf/license.dat`

  Windows example:

  `LSF_LICENSE_FILE= C:\licenses\license.dat`

  or

  `LSF_LICENSE_FILE=\\HostA\licenses\license.dat`

- For a permanent license, the name of the license server host and TCP port number used by the `lmgrd` daemon, in the format *port@host_name*. For example:

  `LSF_LICENSE_FILE="1700@hostD"`

- For a license with redundant servers, use a colon to separate the *port@host_name*s. For example:

  `LSF_LICENSE_FILE="port@hostA:port@hostB:port@hostC"`

  The port number must be the same as that specified in the `SERVER` line of the license file.

Multiple license files should be quoted and must be separated by a pipe character (|).

Windows example:

`LSF_LICENSE_FILE="C:\licenses\license1|C:\licenses\license2|D:\mydir\license3"`

Multiple files may be kept in the same directory, but each one must reference a different license server. When checking out a license, LSF searches the servers in the order in which they are listed, so it checks the second server when there are no more licenses available from the first server.

If this parameter is not defined, LSF assumes the default location.

Default If you installed LSF with a default installation, the license file is installed in the LSF configuration directory (`LSF_CONFDIR/license.dat`).

If you installed LSF with a custom installation, you specify the license installation directory. The default is the LSF configuration directory (LSF_SERVERDIR for the custom installation).

If you installed FLEXnet separately from LSF to manage other software licenses, the default FLEXnet installation puts the license file in the following location:

- UNIX: `/usr/share/flexlm/licenses/license.dat`
- Windows: `C:\flexlm\license.dat`

LSF_LICENSE_NOTIFICATION_INTERVAL

Syntax **LSF_LICENSE_NOTIFICATION_INTERVAL=***hours*

Description Specifies how often notification email is sent to the primary cluster administrator about overuse of LSF Family product licenses and LSF License Scheduler tokens.

Recommended value To avoid getting the same audit information more than once, set LSF_LICENSE_NOTIFICATION_INTERVAL greater than 24 hours.

Example notification email

```
Subject: LSF license overuse

LSF Administrator:
Your cluster has experienced license overuse.

Platform Product License Name: LSF_MANAGER
CLASS E license usage: 0 in total; 8 in use (8 overused).
Overuse Hosts:  hostA

Use lim -t and lshosts -l or see
/usr/opt/lsf6.2/log/lsf.cluster_6.2.license.acct file for
details.

Please contact Platform Support at support@platform.com for
information about getting additional licenses.
```

Default 24 hours

See also

- LSF_LICENSE_ACCT_PATH
- LSF_LOGDIR
- `lsf.cluster_name.license.acct`
- `bld.license.acct`

LSF_LIM_DEBUG

Syntax **`LSF_LIM_DEBUG=1 | 2`**

Description Sets LSF to debug mode.

If LSF_LIM_DEBUG is defined, LIM operates in single user mode. No security checking is performed, so LIM should not run as root.

LIM will not look in the services database for the LIM service port number. Instead, it uses port number 36000 unless LSF_LIM_PORT has been defined.

Specify 1 for this parameter unless you are testing LSF.

Valid Values

- LSF_LIM_DEBUG=1
  LIM runs in the background with no associated control terminal.
- LSF_LIM_DEBUG=2
  LIM runs in the foreground and prints error messages to `tty`.

Default Undefined

See also LSF_RES_DEBUG, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_LIM_IGNORE_CHECKSUM

Syntax **`LSF_LIM_IGNORE_CHECKSUM=y | Y`**

Description Configure LSF_LIM_IGNORE_CHECKSUM=Y to ignore warning messages logged to lim log files on non-master hosts.

When LSF_MASTER_LIST is set, `lsadmin reconfig` only restarts master candidate hosts (for example, after adding or removing hosts from the cluster). This can cause superflous warning messages like the following to be logged in the lim log files for non-master hosts because lim on these hosts are not restarted after configuration change:

```
Aug 26 13:47:35 2005 9746 4 6.2 xdr_loadvector: Sender
<10.225.36.46:9999> has a different configuration
```

Default Undefined

See also LSF_MASTER_LIST

LSF_LIM_PLUGINDIR

Syntax **`LSF_LIM_PLUGINDIR=`***path*

Description The path to `liblimvcl.so`. Used only with SUN HPC.

Default Path to LSF_LIBDIR

See also LSF_RES_PLUGINDIR

LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT

Syntax Example: **`LSF_LIM_PORT=`***port_number*

Description TCP service ports to use for communication with the LSF daemons.

If port parameters are undefined, LSF obtains the port numbers by looking up the LSF service names in the `/etc/services` file or the NIS (UNIX). If it is not possible to modify the services database, you can define these port parameters to set the port numbers.

With careful use of these settings along with the LSF_ENVDIR and PATH environment variables, it is possible to run two versions of the LSF software on a host, selecting between the versions by setting the PATH environment variable to include the correct version of the commands and the LSF_ENVDIR environment variable to point to the directory containing the appropriate `lsf.conf` file.

Default On UNIX, the default is to get port numbers from the services database.

On Windows, these parameters are mandatory.

Default port number values are:

- LSF_LIM_PORT=6879
- LSF_RES_PORT=6878
- LSB_MBD_PORT=6881
- LSB_SBD_PORT=6882

LSF_LIM_SOL27_PLUGINDIR

Syntax **`LSF_LIM_SOL27_PLUGINDIR=`***path*

Description The path to `liblimvcl.so`. Used only with Solaris2.7.

Default Path to LSF_LIBDIR

See also LSF_RES_SOL27_PLUGINDIR

LSF_LOCAL_RESOURCES

Syntax **LSF_LOCAL_RESOURCES="***resource ...***"**

Description Defines instances of local resources residing on the slave host.

- For numeric resources, defined name-value pairs:

  "[resourcemap *value*******resource_name***]"**

- For Boolean resources, the value will be the resource name in the form:

  "[resource *resource_name***]"**

When the slave host calls the master host to add itself, it also reports its local resources. The local resources to be added must be defined in `lsf.shared`.

If the same resource is already defined in `lsf.shared` as `default` or `all`, it cannot be added as a local resource. The shared resource overrides the local one.

LSF_LOCAL_RESOURCES is usually set in the `slave.config` file during installation. If LSF_LOCAL_RESOURCES are already defined in a local `lsf.conf` on the slave host, `lsfinstall` *does not* add resources you define in LSF_LOCAL_RESOURCES in `slave.config`. You should not have duplicate LSF_LOCAL_RESOURCES entries in `lsf.conf`. If local resources are defined more than once, only the last definition is valid.

IMPORTANT **Resources must already be mapped to hosts in the ResourceMap section of lsf.cluster.*cluster_name*. If the ResourceMap section does not exist, local resources are not added.**

Example `LSF_LOCAL_RESOURCES="[resourcemap 1*verilog] [resource linux]"`

Default Undefined

LSF_LOG_MASK

Syntax **LSF_LOG_MASK=***message_log_level*

Description Specifies the logging level of error messages for LSF daemons.

For example:

```
LSF_LOG_MASK=LOG_DEBUG
```

To specify the logging level of error messages, use LSB_CMD_LOG_MASK. To specify the logging level of error messages for LSF commands, use LSF_CMD_LOG_MASK.

On UNIX, this is similar to `syslog`. All messages logged at the specified level or higher are recorded; lower level messages are discarded. The LSF_LOG_MASK value can be any log priority symbol that is defined in `syslog.h` (see `syslog`(8)).

The log levels in order from highest to lowest are:

- LOG_EMERG
- LOG_ALERT

- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

The most important LSF log messages are at the LOG_ERR or LOG_WARNING level. Messages at the LOG_INFO and LOG_DEBUG level are only useful for debugging.

Although message log level implements similar functionalities to UNIX `syslog`, there is no dependency on UNIX `syslog`. It works even if messages are being logged to files instead of `syslog`.

LSF logs error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSF_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

In versions earlier than LSF 4.0, you needed to restart the daemons after setting LSF_LOG_MASK in order for your changes to take effect.

LSF 4.0 implements dynamic debugging, which means you do not need to restart the daemons after setting a debugging environment variable.

Default LOG_WARNING

See also LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_DEBUG_NQS, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_DEBUG_LIM, LSB_DEBUG_MBD, LSF_DEBUG_RES, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSF_LOG_MASK_WIN

Syntax **`LSF_LOG_MASK_WIN=`***message_log_level*

Description Allows you to reduce the information logged to the LSF Windows event log files. Messages of lower severity than the specified level are discarded.

For all LSF files, the types of messages saved depends on LSF_LOG_MASK, so the threshold for the Windows event logs is either LSF_LOG_MASK or LSF_LOG_MASK_WIN, whichever is higher. LSF_LOG_MASK_WIN is ignored if LSF_LOG_MASK is set to a higher level.

The LSF event log files for Windows are:

- `lim.log.`*host_name*
- `res.log.`*host_name*
- `sbatchd.log.`*host_name*
- `mbatchd.log.`*host_name*
- `pim.log.`*host_name*

The log levels you can specify for this parameter, in order from highest to lowest, are:

- LOG_ERR
- LOG_WARNING
- LOG_INFO
- LOG_NONE (LSF does not log Windows events)

Default LOG_ERR

See also LSF_LOG_MASK

LSF_LOGDIR

Syntax **`LSF_LOGDIR=`***dir*

Description Defines the LSF system log file directory. Error messages from all servers are logged into files in this directory. To effectively use debugging, set LSF_LOGDIR to a directory such as `/tmp`. This can be done in your own environment from the shell or in `lsf.conf`.

Windows LSF_LOGDIR is required on Windows if you wish to enable logging.

You also need to define LSF_LOGDIR_USE_WIN_REG=n. If you define LSF_LOGDIR without defining LSF_LOGDIR_USE_WIN_REG=n, LSF logs error messages into files in the default local directory specified in the following registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Platform Computing
Corporation\LSF\cluster_name\LSF_LOGDIR
```

If a server is unable to write in the LSF system log file directory, LSF attempts to write to the following directories in the following order:

- LSF_TMPDIR if defined
- `%TMP%` if defined
- `%TEMP%` if defined
- System directory, for example, `c:\winnt`

UNIX If a server is unable to write in this directory, the error logs are created in `/tmp` on UNIX.

If LSF_LOGDIR is not defined, `syslog` is used to log everything to the system log using the LOG_DAEMON facility. The syslog facility is available by default on most UNIX systems. The `/etc/syslog.conf` file controls the way messages are logged and the files they are logged to. See the man pages for the `syslogd` daemon and the `syslog` function for more information.

Default Undefined

On UNIX, if undefined, log messages go to `syslog`.

On Windows, if undefined, no logging is performed.

See also LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR_USE_WIN_REG, LSF_TIME_CMD

Files

- `lim.log.`*host_name*
- `res.log.`*host_name*
- `sbatchd.log.`*host_name*
- `sbatchdc.log.host_name` (Windows only)
- `mbatchd.log.`*host_name*
- `eeventd.log.`*host_name*
- `pim.log.`*host_name*

LSF_LOGDIR_USE_WIN_REG

Syntax **`LSF_LOGDIR_USE_WIN_REG=n | N`**

Description Windows only.

If set, LSF logs error messages into files in the directory specified by LSF_LOGDIR in `lsf.conf`.

Use this parameter to enable LSF to save log files in a different location from the default local directory specified in the Windows registry.

If not set, or if set to any value other than N or n, LSF logs error messages into files in the default local directory specified in the following Windows registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Platform Computing
Corporation\LSF\cluster_name\LSF_LOGDIR
```

Default Not set.

LSF uses the default local directory specified in the Windows registry.

See also LSF_LOGDIR

LSF_MACHDEP

Syntax **`LSF_MACHDEP=`***dir*

Description Specifies the directory in which machine-dependent files are installed. These files cannot be shared across different types of machines.

In clusters with a single host type, LSF_MACHDEP is usually the same as LSF_INDEP. The machine dependent files are the user commands, daemons, and libraries. You should not need to modify this parameter.

As shown in the following list, LSF_MACHDEP is incorporated into other LSF variables.

- LSF_BINDIR=$LSF_MACHDEP/bin
- LSF_LIBDIR=$LSF_MACHDEP/lib
- LSF_SERVERDIR=$LSF_MACHDEP/etc
- XLSF_UIDDIR=$LSF_MACHDEP/lib/uid

Default `/usr/share/lsf`

See also LSF_INDEP

LSF_MANDIR

Syntax **LSF_MANDIR=***dir*

Description Directory under which all man pages are installed.

The man pages are placed in the `man1`, `man3`, `man5`, and `man8` subdirectories of the LSF_MANDIR directory. This is created by the LSF installation process, and you should not need to modify this parameter.

Man pages are installed in a format suitable for BSD-style `man` commands.

For most versions of UNIX, you should add the directory LSF_MANDIR to your MANPATH environment variable. If your system has a `man` command that does not understand MANPATH, you should either install the man pages in the `/usr/man` directory or get one of the freely available `man` programs.

Default `LSF_INDEP/man`

LSF_MASTER_LIST

Syntax **LSF_MASTER_LIST="***host_name ...***"**

Description Optional. Defines a list of hosts that are candidates to become the master host for the cluster.

Listed hosts must be defined in `lsf.cluster.`*cluster_name*.

Host names are separated by spaces.

When you run `lsadmin reconfig` to reconfigure the cluster, only the master LIM candidates read `lsf.shared` and `lsf.cluster.`*cluster_name* to get updated information. The elected master LIM sends configuration information to slave LIMs.

Master host candidates should share LSF configuration and binaries.

To dynamically add or remove hosts, you must define LSF_MASTER_LIST.

If you have a large number of non-master hosts, you should configure LSF_LIM_IGNORE_CHECKSUM=Y to ignore warning messages like the following logged to lim log files on non-master hosts.

```
Aug 26 13:47:35 2005 9746 4 6.2 xdr_loadvector: Sender
<10.225.36.46:9999> has a different configuration
```

Default Undefined

See also LSF_LIM_IGNORE_CHECKSUM

LSF_MC_NON_PRIVILEGED_PORTS

Syntax **LSF_MC_NON_PRIVILEGED_PORTS=y | Y**

Description MultiCluster only. If this parameter is enabled in one cluster, it must be enabled in all clusters.

Specify Y to make LSF daemons use non-privileged ports for communication across clusters.

Compatibility This disables privileged port daemon authentication, which is a security feature. If security is a concern, you should use `eauth` for LSF daemon authentication (see **LSF_AUTH_DAEMONS** in `lsf.conf`).

Default Undefined (LSF daemons use privileged port authentication)

LSF_MISC

Syntax **LSF_MISC=***dir*

Description Directory in which miscellaneous machine independent files, such as example source programs and scripts, are installed.

Default `LSF_CONFDIR/misc`

LSF_NON_PRIVILEGED_PORTS

Syntax **LSF_NON_PRIVILEGED_PORTS=y | Y**

Description Disables privileged ports usage.

By default, LSF daemons and clients running under root account will use privileged ports to communicate with each other. Without LSF_NON_PRIVILEGED_PORTS defined, and if LSF_AUTH is not defined in `lsf.conf`, LSF daemons check privileged port of request message to do authentication.

If LSF_NON_PRIVILEGED_PORTS=Y is defined, LSF clients (LSF commands and daemons) will not use privileged ports to communicate with daemons and LSF daemons will not check privileged ports of incoming requests to do authentication.

LSF_NIOS_DEBUG

Syntax **LSF_NIOS_DEBUG=1**

Description Turns on NIOS debugging for interactive jobs.

If LSF_NIOS_DEBUG=1, NIOS debug messages are written to standard error.

This parameter can also be defined as an environment variable.

When LSF_NIOS_DEBUG and LSF_CMD_LOGDIR are defined, NIOS debug messages are logged in `nios.log.`*`host_name`*. in the location specified by LSF_CMD_LOGDIR.

If LSF_NIOS_DEBUG is defined, and the directory defined by LSF_CMD_LOGDIR is inaccessible, NIOS debug messages are logged to `/tmp/nios.log.`*`host_name`* instead of `stderr`.

On Windows, NIOS debug messages are also logged to the temporary directory.

Default Undefined

See also LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_NIOS_JOBSTATUS_INTERVAL

Syntax **LSF_NIOS_JOBSTATUS_INTERVAL=***time_minutes*

Description Applies only to interactive batch jobs.

Time interval at which NIOS polls mbatchd to check if a job is still running. Used to retrieve a job's exit status in the case of an abnormal exit of NIOS, due to a network failure for example.

Use this parameter if you run interactive jobs and you have scripts that depend on an exit code being returned.

When this parameter is undefined and a network connection is lost, mbatchd cannot communicate with NIOS and the return code of a job is not retrieved.

When this parameter is defined, before exiting, NIOS polls mbatchd on the interval defined by LSF_NIOS_JOBSTATUS_INTERVAL to check if a job is still running. NIOS continues to poll mbatchd until it receives an exit code or mbatchd responds that the job does not exist (if the job has already been cleaned from memory for example).

If an exit code cannot be retrieved, NIOS generates an error message and the code -11.

Valid Values Any integer greater than zero

Default Undefined

Notes Set this parameter to large intervals such as 15 minutes or more so that performance is not negatively affected if interactive jobs are pending for too long. NIOS always calls mbatchd on the defined interval to confirm that a job is still pending and this may add load to mbatchd.

See also Environment variable LSF_NIOS_PEND_TIMEOUT

LSF_NIOS_RES_HEARTBEAT

Syntax **LSF_NIOS_RES_HEARTBEAT=***time_minutes*

Description Applies only to interactive non-parallel batch jobs.

Defines how long NIOS waits before sending a message to RES to determine if the connection is still open.

Use this parameter to ensure NIOS exits when a network failure occurs instead of waiting indefinitely for notification that a job has been completed. When a network connection is lost, RES cannot communicate with NIOS and as a result, NIOS does not exit.

When this parameter is defined, if there has been no communication between RES and NIOS for the defined period of time, NIOS sends a message to RES to see if the connection is still open. If the connection is no longer available, NIOS exits.

Valid values Any integer greater than zero

Default Undefined

Notes The time you set this parameter to depends how long you want to allow NIOS to wait before exiting. Typically, it can be a number of hours or days. Too low a number may add load to the system.

LSF_PAM_HOSTLIST_USE

Syntax **LSF_PAM_HOSTLIST_USE=unique**

Description Used to start applications that use both OpenMP and MPI.

Valid values `unique`

Default Undefined

Notes At job submission, LSF reserves the correct number of processors and PAM will start only 1 process per host. For example, to reserve 32 processors and run on 4 processes per host, resulting in the use of 8 hosts:

```
% bsub -n 32 -R "span[ptile=4]" pam yourOpenMPJob
```

Where defined This parameter can alternatively be set as an environment variable. For example:

```
setenv LSF_PAM_HOSTLIST_USE unique
```

LSF_PAM_PLUGINDIR

Syntax **LSF_PAM_PLUGINDIR=***path*

Description The path to `libpamvcl.so`. Used with Platform LSF HPC.

Default Path to LSF_LIBDIR

See also LSF_RES_PLUGINDIR

LSF_PAM_USE_ASH

Syntax **LSF_PAM_USE_ASH=y** | **Y**

Description Enables LSF to use the SGI IRIX Array Session Handles (ASH) to propagate signals to the parallel jobs.

See the IRIX system documentation and the `array_session`(5) man page for more information about array sessions.

Default Undefined

LSF_POE_TIMEOUT_BIND

Syntax **LSF_POE_TIMEOUT_BIND=***seconds*

Description Specifies the time in seconds for the `poe_w` wrapper to keep trying to set up a server socket to listen on.

`poe_w` is the wrapper for the IBM `poe` driver program.

LSF_POE_TIMEOUT_BIND can also be set as an environment variable for `poe_w` to read.

Default 120 seconds

LSF_POE_TIMEOUT_SELECT

Syntax **LSF_POE_TIMEOUT_SELECT=***seconds*

Description Specifies the time in seconds for the poe_w wrapper to wait for connections from the pmd_w wrapper. pmd_w is the wrapper for pmd (IBM PE Partition Manager Daemon).

LSF_POE_TIMEOUT_SELECT can also be set as an environment variable for poe_w to read.

Default 160 seconds

LSF_PIM_INFODIR

Syntax **LSF_PIM_INFODIR=***path*

Description The path to where PIM writes the pim.info.*host_name* file.

Specifies the path to where the process information is stored. The process information resides in the file pim.info.*host_name*. The PIM also reads this file when it starts so that it can accumulate the resource usage of dead processes for existing process groups.

Default Undefined. If undefined, the system uses /tmp.

LSF_PIM_SLEEPTIME

Syntax **LSF_PIM_SLEEPTIME=***time_seconds*

Description The reporting period for PIM.

PIM updates the process information every 15 minutes unless an application queries this information. If an application requests the information, PIM will update the process information every LSF_PIM_SLEEPTIME seconds. If the information is not queried by any application for more than 5 minutes, the PIM will revert back to the 15 minute update period.

Default 15

LSF_PIM_SLEEPTIME_UPDATE

Syntax **LSF_PIM_SLEEPTIME_UPDATE=y | n**

Description UNIX only.

Use this parameter to improve job throughput and reduce a job's start time if there are many jobs running simultaneously on a host. This parameter reduces communication traffic between sbatchd and PIM on the same host.

When this parameter is undefined or set to n, sbatchd queries PIM as needed for job process information.

When this parameter is defined, sbatchd does not query PIM immediately as it needs information—sbatchd will only query PIM every LSF_PIM_SLEEPTIME seconds.

Limitations When this parameter is defined:

- sbatchd may be intermittently unable to retrieve process information for jobs whose run time is smaller than LSF_PIM_SLEEPTIME.

- It may take longer to view resource usage with `bjobs -l`.

Default Undefined

LSF_RES_ACCT

Syntax **LSF_RES_ACCT=***time_milliseconds* | **0**

Description If this parameter is defined, RES will log information for completed and failed tasks by default (see `lsf.acct`(5)).

The value for LSF_RES_ACCT is specified in terms of consumed CPU time (milliseconds). Only tasks that have consumed more than the specified CPU time will be logged.

If this parameter is defined as LSF_RES_ACCT=0, then all tasks will be logged.

For those tasks that consume the specified amount of CPU time, RES generates a record and appends the record to the task log file `lsf.acct.`*`host_name`*. This file is located in the LSF_RES_ACCTDIR directory.

If this parameter is not defined, the LSF administrator must use the `lsadmin` command (see `lsadmin`(8)) to turn task logging on after RES has started.

Default Undefined

See also LSF_RES_ACCTDIR

LSF_RES_ACCTDIR

Syntax **LSF_RES_ACCTDIR=***dir*

Description The directory in which the RES task log file `lsf.acct.`*`host_name`* is stored.

If LSF_RES_ACCTDIR is not defined, the log file is stored in the `/tmp` directory.

Default (UNIX)`/tmp`

(Windows) `C:\temp`

See also LSF_RES_ACCT

LSF_RES_ACTIVE_TIME

Syntax **LSF_RES_ACTIVE_TIME=***seconds*

Description Time in seconds before LIM reports that RES is down.

Minimum value 10 seconds

Default 90 seconds

LSF_RES_CONNECT_RETRY

Syntax **LSF_RES_CONNECT_RETRY=***integer* | **0**

description The number of attempts by RES to reconnect to NIOS.

If LSF_RES_CONNECT_RETRY is not defined, the default value is used.

Default 0

See Also LSF_NIOS_RES_HEARTBEAT

LSF_RES_DEBUG

Syntax **LSF_RES_DEBUG=1 | 2**

Description Sets RES to debug mode.

If LSF_RES_DEBUG is defined, the Remote Execution Server (RES) will operate in single user mode. No security checking is performed, so RES should not run as root. RES will not look in the services database for the RES service port number. Instead, it uses port number 36002 unless LSF_RES_PORT has been defined.

Specify 1 for this parameter unless you are testing RES.

Valid values

- LSF_RES_DEBUG=1
  RES runs in the background with no associated control terminal.
- LSF_RES_DEBUG=2
  RES runs in the foreground and prints error messages to `tty`.

Default Undefined

See also LSF_LIM_DEBUG, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_RES_PLUGINDIR

Syntax **LSF_RES_PLUGINDIR=***path*

Description The path to `lsbresvcl.so`. Used only with SUN HPC.

Default Path to LSF_LIBDIR

See also LSF_PAM_PLUGINDIR, LSF_LIM_PLUGINDIR

LSF_RES_PORT

See "LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT" on page 555.

LSF_RES_RLIMIT_UNLIM

Syntax **LSF_RES_RLIMIT_UNLIM=cpu | fsize | data | stack | core | vmem**

Description By default, RES sets the hard limits for a remote task to be the same as the hard limits of the local process. This parameter specifies those hard limits which are to be set to unlimited, instead of inheriting those of the local process.

Valid values are `cpu`, `fsize`, `data`, `stack`, `core`, and `vmem`, for CPU, file size, data size, stack, core size, and virtual memory limits, respectively.

Example The following example sets the CPU, core size, and stack hard limits to be unlimited for all remote tasks:

```
LSF_RES_RLIMIT_UNLIM="cpu core stack"
```

Default Undefined

See also LSF_LIM_SOL27_PLUGINDIR

LSF_RES_SOL27_PLUGINDIR

Syntax **LSF_RES_SOL27_PLUGINDIR=***path*

Description The path to `libresvcl.so`. Used only used with Solaris2.7.

If you want to link a 64-bit object with RES, then you should set LSF_RES_SOL27_PLUGINDIR.

Default Path to LSF_LIBDIR

LSF_RES_TIMEOUT

Syntax **LSF_RES_TIMEOUT=***time_seconds*

Description Timeout when communicating with RES.

Default 15

LSF_ROOT_REX

Syntax **LSF_ROOT_REX=local**

Description UNIX only.

Allows root remote execution privileges (subject to identification checking) on remote hosts, for both interactive and batch jobs. Causes RES to accept requests from the superuser (root) on remote hosts, subject to identification checking.

If LSF_ROOT_REX is undefined, remote execution requests from user root are refused.

Theory Sites that have separate root accounts on different hosts within the cluster should not define LSF_ROOT_REX. Otherwise, this setting should be based on local security policies.

The `lsf.conf` file is host-type specific and not shared across different platforms. You must make sure that `lsf.conf` for all your host types are changed consistently.

Default Undefined (root execution is not allowed)

See also LSF_TIME_CMD, LSF_AUTH

LSF_RSH

Syntax **LSF_RSH=***command* [*command_opions*]

Description Specifies shell commands to use when the following LSF commands require remote execution:

- `badmin hstartup`
- `bpeek`
- `lsadmin limstartup`
- `lsadmin resstartup`
- `lsfrestart`
- `lsfshutdown`
- `lsfstartup`
- `lsrcp`

By default, `rsh` is used for these commands. Use LSF_RSH to enable support for `ssh`.

Default Undefined

Example To use an `ssh` command before trying `rsh` for LSF commands, specify:

```
LSF_RSH="ssh -o 'PasswordAuthentication no' -o 'StrictHostKeyChecking no'"
```

`ssh` options such as `PasswordAuthentication` and `StrictHostKeyChecking` can also be configured in the global `SSH_ETC/ssh_config` file or `$HOME/.ssh/config`.

See also `ssh`(1) `ssh_config`(5)

LSF_SECUREDIR

Syntax **`LSF_SECUREDIR=`***path*

Description Windows only; mandatory if using `lsf.sudoers`.

Path to the directory that contains the file `lsf.sudoers` (shared on an NTFS file system).

LSF_SERVER_HOSTS

Syntax **`LSF_SERVER_HOSTS="`***host_name ...***`"`**

Description Defines one or more server hosts that the application should contact to find a Load Information Manager (LIM). This is used on client hosts on which no LIM is running on the local host. LSF server hosts are hosts that run LSF daemons and provide loading-sharing services. Client hosts are hosts that only run LSF commands or applications but do not provide services to any hosts.

If LSF_SERVER_HOSTS is not defined, the application tries to contact the LIM on the local host.

The host names in LSF_SERVER_HOSTS must be enclosed in quotes and separated by white space. For example:

```
LSF_SERVER_HOSTS="hostA hostD hostB"
```

The length of the parameter string must be less then 4096 characters.

Default Undefined

LSF_SERVERDIR

Syntax **`LSF_SERVERDIR=`***dir*

Description Directory in which all server binaries and shell scripts are installed.

These include `lim`, `res`, `nios`, `sbatchd`, `mbatchd`, and `mbschd`. If you use `elim`, `eauth`, `eexec`, `esub`, etc, they are also installed in this directory.

Default `LSF_MACHDEP/etc`

See also LSB_ECHKPNT_METHOD_DIR

LSF_SHELL_AT_USERS

Syntax **`LSF_SHELL_AT_USERS="`***user_name user_name ...***`"`**

Description Applies to `lstcsh` only. Specifies users who are allowed to use `@` for host redirection. Users not specified with this parameter cannot use host redirection in `lstcsh`.

If this parameter is undefined, all users are allowed to use `@` for host redirection in `lstcsh`.

Default Undefined

LSF_SHIFT_JIS_INPUT

Syntax **LSF_SHIFT_JIS_INPUT=y | n**

Description Enables LSF to accept Shift-JIS character encoding for job information (for example, user names, queue names, job names, job group names, project names, commands and arguments, esub parameters, external messages, etc.)

Default `n`

LSF_SLURM_DISABLE_CLEANUP

Syntax **LSF_SLURM_DISABLE_CLEANUP=y | Y**

Description Disables cleanup of non-LSF jobs running in a SLURM LSF partition on an HP XC machine.

By default, only LSF jobs are allowed to run within a SLURM LSF partition. LSF periodically cleans up any jobs submitted outside of LSF. This clean up period is defined through LSB_RLA_UPDATE.

For example, the following `srun` job is not submitted through LSF, so it is terminated:

```
% srun -n 4 -p lsf sleep 100000
srun: error: n13: task[0-1]: Terminated
srun: Terminating job
```

If LSF_SLURM_DISABLE_CLEANUP=Y is set, this job would be allowed to run.

Default Undefined

LSF_SLURM_TMPDIR

Syntax **LSF_SLURM_TMPDIR=***path*

Description Specifies the LSF HPC tmp directory for HP XC machines. The default LSF_TMPDIR `/tmp` cannot be shared across nodes, so LSF_SLURM_TMPDIR must specify a path that is accessible on all XC nodes.

Default `/hptc_cluster/lsf/tmp`

LSF_STRICT_CHECKING

Syntax **LSF_STRICT_CHECKING=Y**

Description If set, enables more strict checking of communications between LSF daemons and between LSF commands and daemons when LSF is used in an untrusted environment, such as a public network like the Internet.

If you enable this parameter, you must enable it in the entire cluster, as it affects all communications within LSF. If it is used in a MultiCluster environment, it must be enabled in all clusters, or none. Ensure that all binaries and libraries are upgraded to LSF Version 6.2, including LSF_BINDIR, LSF_SERVERDIR and LSF_LIBDIR directories, if you enable this parameter.

If your site uses any programs that use the LSF base and batch APIs, or LSF MPI (Message Passing Interface), they need to be recompiled using the LSF Version 6.2 APIs before they can work properly with this option enabled.

IMPORTANT **You must shut down the entire cluster before enabling or disabling this parameter.**

If LSF_STRICT_CHECKING is defined, and your cluster has slave hosts that are dynamically added, LSF_STRICT_CHECKING must be configured in the local lsf.conf on all slave hosts.

Valid value Set to `Y` to enable this feature.

Default Undefined. LSF is secure in trusted environments.

LSF_STRIP_DOMAIN

Syntax **`LSF_STRIP_DOMAIN=`***domain_suffix* [**`:`***domain_suffix* ...]

Description (Optional) If all of the hosts in your cluster can be reached using short host names, you can configure LSF to use the short host names by specifying the portion of the domain name to remove. If your hosts are in more than one domain or have more than one domain name, you can specify more than one domain suffix to remove, separated by a colon (:).

For example, given this definition of LSF_STRIP_DOMAIN,

`LSF_STRIP_DOMAIN=.foo.com:.bar.com`

LSF accepts `hostA`, `hostA.foo.com`, and `hostA.bar.com` as names for host `hostA`, and uses the name `hostA` in all output. The leading period '.' is required.

Example:

`LSF_STRIP_DOMAIN=.platform.com:.generic.com`

In the above example, LSF accepts `hostA`, `hostA.platform.com`, and `hostA.generic.com` as names for `hostA`, and uses the name `hostA` in all output.

Setting this parameter only affects host names displayed through LSF, it does not affect DNS host lookup.

Default Undefined

LSF_TIME_CMD

Syntax **`LSF_TIME_CMD=`***timimg_level*

Description The timing level for checking how long LSF commands run. Time usage is logged in milliseconds; specify a positive integer.

Default Undefined

See also LSB_TIME_MBD, LSB_TIME_SBD, LSB_TIME_CMD, LSF_TIME_LIM, LSF_TIME_RES

LSF_TIME_LIM

Syntax **LSF_TIME_LIM=***timing_level*

Description The timing level for checking how long LIM routines run.

Time usage is logged in milliseconds; specify a positive integer.

Default Undefined

See also LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_RES

LSF_TIME_RES

Syntax **LSF_TIME_RES=***timing_level*

Description The timing level for checking how long RES routines run.

Time usage is logged in milliseconds; specify a positive integer.

Default Undefined

See also LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM

LSF_TMPDIR

Syntax **LSF_TMPDIR=***dir*

Description Specifies the path and directory for temporary job output.

When LSF_TMPDIR is defined in `lsf.conf`, LSF creates a temporary directory under the directory specified by LSF_TMPDIR on the execution host when a job is started and sets the temporary directory environment variable for the job.

When LSF_TMPDIR is defined as an environment variable, it overrides the LSF_TMPDIR specified in `lsf.conf`. LSF removes the temporary directory and the files that it contains when the job completes.

The name of the temporary directory has the following format:

```
$LSF_TMPDIR/job_ID.tmpdir
```

On UNIX, the directory has the permission 0700.

After adding LSF_TMPDIR to `lsf.conf`, use `badmin hrestart all` to reconfigure your cluster.

This parameter can also be specified from the command line.

Valid values Specify any valid path up to a maximum length of 256 characters. The 256 character maximum path length includes the temporary directories and files that the system creates as jobs run. The path that you specify for LSF_TMPDIR should be as short as possible to avoid exceeding this limit.

UNIX Specify an absolute path. For example:

```
LSF_TMPDIR=/usr/share/lsf_tmp
```

Windows Specify a UNC path or a path with a drive letter. For example:

```
LSF_TMPDIR=\\HostA\temp\lsf_tmpor
LSF_TMPDIR=D:\temp\lsf_tmp
```

Default By default, LSF_TMPDIR is not enabled. If LSF_TMPDIR is not specified either in the environment or in `lsf.conf`, this parameter is defined as follows:

- On UNIX: $TMPDIR or `/tmp`
- On Windows: %TMP%, %TEMP, or %SystemRoot%

LSF_TOPD_PORT

Syntax **LSF_TOPD_PORT=***port_number*

Description UDP port used for communication between the LSF cpuset topology daemon (`topd`) and the cpuset ELIM. Used with SGI IRIX cpuset support.

Default Undefined

LSF_TOPD_WORKDIR

Syntax **LSF_TOPD_WORKDIR=***directory*

Description Directory to store the IRIX cpuset permission file and the event file for the cpuset topology daemon (`topd`). Used with SGI IRIX cpuset support.

You should avoid using `/tmp` or any other directory that is automatically cleaned up by the system. Unless your installation has restrictions on the LSB_SHAREDIR directory, you should use the default for LSF_TOPD_WORKDIR.

Default `LSB_SHAREDIR/topd_dir.port_number`

Where *port_number* is the value you set for LSF_TOPD_PORT.

LSF_ULDB_DOMAIN

Syntax **LSF_ULDB_DOMAIN="***domain_name ...***"**

Description LSF_ULDB_DOMAIN specifies the name of the LSF domain in the ULDB domain directive. A domain definition of name *domain_name* must be configured in the IRIX `jlimit.in` input file.

Used with IRIX User Limits Database (ULDB). Configures LSF so that jobs submitted to a host with the IRIX job limits option installed are subject to the job limits configured in the IRIX User Limits Database (ULDB).

The ULDB contains job limit information that system administrators use to control access to a host on a per user basis. The job limits in the ULDB override the system default values for both job limits and process limits. When a ULDB domain is configured, the limits will be enforced as IRIX job limits.

If the ULDB domain specified in LSF_ULDB_DOMAIN is not valid or does not exist, LSF uses the limits defined in the domain named `batch`. If the `batch` domain does not exist, then the system default limits are set.

When an LSF job is submitted, an IRIX job is created, and the job limits in the ULDB are applied.

Next, LSF resource usage limits are enforced for the IRIX job under which the LSF job is running. LSF limits override the corresponding IRIX job limits. The ULDB limits are used for any LSF limits that are not defined. If the job reaches the IRIX job limits, the action defined in the IRIX system is used.

IRIX job limits in the ULDB apply only to batch jobs.

See the IRIX 6.5.8 resource administration documentation for information about configuring ULDB domains in the `jlimit.in` file.

LSF resource usage limits controlled by ULDB

- PROCESSLIMIT—Corresponds to IRIX JLIMIT_NUMPROC; `fork`(2) fails, but the existing processes continue to run
- MEMLIMIT—Corresponds to JLIMIT_RSS; Resident pages above the limit become prime swap candidates
- DATALIMIT—Corresponds to LIMIT_DATA; malloc(3) calls in the job fail with errno set to ENOMEM
- CPULIMIT—Corresponds to JLIMIT_CPU; IRIX sends SIGXCPU signal to job, then after the grace period expires, sends SIGINT, SIGTERM, and SIGKILL
- FILELIMIT—No corresponding IRIX limit; use process limit RLIMIT_FSIZE
- STACKLIMIT—No corresponding IRIX limit; use process limit RLIMIT_STACK
- CORELIMIT—No corresponding IRIX limit; use process limit RLIMIT_CORE
- SWAPLIMIT—Corresponds to JLIMIT_VMEM; use process limit RLIMIT_VMEM

Increasing the default MEMLIMIT for ULDB

In some pre-defined LSF queues, such as `normal`, the default MEMLIMIT is set to 5000 (5 MB). However, if ULDB is enabled (LSF_ULDB_DOMAIN is defined) the MEMLIMIT should be set greater than 8000 in `lsb.queues`.

Example ULDB domain configuration

The following steps enable the ULDB domain LSF for user `user1`:

1. Define the LSF_ULDB_DOMAIN parameter in `lsf.conf`:

   ```
   ...
   LSF_ULDB_DOMAIN=LSF
   ...
   ```

   Note that you can set the LSF_ULDB_DOMAIN to include more than one domain. For example:

   ```
   LSF_ULDB_DOMAIN="lsf:batch:system"
   ```

2. Configure the domain directive `LSF` in the `jlimit.in` file:

   ```
   domain <LSF> {                              # domain for LSF
           jlimit_numproc_cur=unlimited
           jlimit_numproc_max=unlimited      # JLIMIT_NUMPROC
           jlimit_nofile_cur=unlimited
           jlimit_nofile_max=unlimited       # JLIMIT_NOFILE
           jlimit_rss_cur=unlimited
           jlimit_rss_max=unlimited          # JLIMIT_RSS
           jlimit_vmem_cur=128M
   ```

```
            jlimit_vmem_max=256M           # JLIMIT_VMEM
            jlimit_data_cur=unlimited
            jlimit_data_max=unlimited       # JLIMIT_DATA
            jlimit_cpu_cur=80
            jlimit_cpu_max=160             # JLIMIT_CPU
    }
```

3 Configure the user limit directive for `user1` in the `jlimit.in` file:

```
user user1 {
        LSF {
           jlimit_data_cur=128M
           jlimit_data_max=256M
         }
}
```

4 Use the IRIX `genlimits` command to create the user limits database:

```
genlimits -l -v
```

Default Undefined

LSF_USE_HOSTEQUIV

Syntax **LSF_USE_HOSTEQUIV=y | Y**

Description (UNIX only; optional)

If LSF_USE_HOSTEQUIV is defined, RES and `mbatchd` call the `ruserok`(3) function to decide if a user is allowed to run remote jobs.

The `ruserok`(3) function checks in the `/etc/hosts.equiv` file and the user's `$HOME/.rhosts` file to decide if the user has permission to execute remote jobs.

If LSF_USE_HOSTEQUIV is not defined, all normal users in the cluster can execute remote jobs on any host.

If LSF_ROOT_REX is set, root can also execute remote jobs with the same permission test as for normal users.

Default Undefined

See also LSF_ROOT_REX

LSF_USER_DOMAIN

Syntax **LSF_USER_DOMAIN=***domain_name* | **.**

Description Set during LSF installation or setup. If you modify this parameter in an existing cluster, you probably have to modify passwords and configuration files also.

Windows or mixed UNIX-Windows clusters only.

Enables default user mapping, and specifies the LSF user domain. The period (`.`) specifies local accounts, not domain accounts.

- A user name specified without a domain is interpreted (on a Windows host) as belonging to the LSF user domain
- A user name specified with the domain name of the LSF user domain is not valid

- In a mixed cluster, this parameter defines a 2-way, 1:1 user map between UNIX user accounts and Windows user accounts belonging to the specified domain, as long as the accounts have the same user name.

  This means jobs submitted by the Windows user account can run on a UNIX host, and jobs submitted by the UNIX account can run on any Windows host that is available to the Windows user account.

If this parameter is undefined, the default user mapping is not enabled. You can still configure user mapping at the user or system level. User account mapping is required to run cross-platform jobs in a UNIX-Windows mixed cluster.

Default

- If you upgrade from LSF 4.0.1 or earlier, the default is the existing LSF user domain.
- For a new, Windows-only cluster, this parameter is undefined (no LSF user domain, no default user mapping).
- For a new, mixed UNIX-Windows cluster, the default is the domain that the Windows installation account belongs to. This can be modified during LSF installation.

LSF_VPLUGIN

Syntax **LSF_VPLUGIN=***path*

Description The full path to the vendor MPI library `libxmpi.so`. Used with Platform LSF HPC.

For PAM to access the SGI MPI `libxmpi.so` library, the file permission mode must be 755 (`-rwxr-xr-x`).

Examples

- HP MPI: `LSF_VPLUGIN=/opt/mpi/lib/pa1.1/libmpirm.sl`
- SGI MPI: `LSF_VPLUGIN=/usr/lib32/libxmpi.so`

Default Undefined

MC_PLUGIN_REMOTE_RESOURCE

Syntax **MC_PLUGIN_REMOTE_RESOURCE=y**

Description MultiCluster job forwarding model only. By default, the submission cluster does not consider remote resources. Define MC_PLUGIN_REMOTE_RESOURCE=y in the submission cluster to allow consideration of remote resources.

Default Undefined. The submission cluster does not consider remote resources.

XLSF_APPDIR

Syntax **XLSF_APPDIR=***dir*

Description (UNIX only; optional) Directory in which X application default files for LSF products are installed.

The LSF commands that use X look in this directory to find the application defaults. Users do not need to set environment variables to use the Platform LSF X applications. The application default files are platform-independent.

Default `LSF_INDEP/misc`

XLSF_UIDDIR

Syntax **XLSF_UIDDIR=***dir*

Description (UNIX only) Directory in which Motif User Interface Definition files are stored.

These files are platform-specific.

Default `LSF_LIBDIR/uid`

lsf.licensescheduler

The `lsf.licensescheduler` file contains Platform LSF License Scheduler configuration information. All sections except ProjectGroup are required.

The command `blinfo` displays configuration information from this file.

Changing lsf.licensescheduler configuration

After making any changes to `lsf.licensescheduler`, run the following commands:

- `bladmin reconfig` to reconfigure `bld`
- `lsadmin reconfig` to reconfigure LIM
- `badmin mbdrestart` to restart `mbatchd`

Contents

- “Parameters Section” on page 578
- “Clusters Section” on page 582
- “ServiceDomain Section” on page 583
- “Feature Section” on page 585
- “ProjectGroup Section” on page 593
- “Projects Section” on page 596

Parameters Section

Description

Required. Defines License Scheduler configuration parameters.

Parameters section structure

The first and last lines are:

```
Begin Parameters
ADMIN=lsadmin
HOSTS=hostA hostB hostC
LMSTAT_PATH=/etc/flexlm/bin
LMSTAT_INTERVAL=30
PORT=9581
End Parameters
```

Each subsequent line describes one configuration parameter. All parameters are mandatory.

Parameters

- "ADMIN"
- "DISTRIBUTION_POLICY_VIOLATION_ACTION"
- "ENABLE_INTERACTIVE"
- "EXT_FILTER_PORT"
- "FLX_LICENSE_FILE"
- "HOSTS"
- "LIB_RECVTIMEOUT"
- "LM_REMOVE_INTERVAL"
- "LM_STAT_INTERVAL"
- "LMSTAT_PATH"
- "LS_MAX_TASKMAN_SESSIONS"
- "PORT"
- "SCHED_INTERVAL"

ADMIN

Syntax

ADMIN=*user_name* ...

Description

Defines the License Scheduler administrator using a valid UNIX user account. You can specify multiple accounts.

DISTRIBUTION_POLICY_VIOLATION_ACTION

Syntax

DISTRIBUTION_POLICY_VIOLATION_ACTION=(PERIOD *reporting_period* **CMD** *reporting_command***)**

- *reporting_period*

  Specify the keyword PERIOD with a positive integer representing the interval (a multiple of LM_STAT_INTERVAL periods) at which License Scheduler checks for distribution policy violations.

- *reporting_command*

  Specify the keyword CMD with the directory path and command that License Scheduler runs when reporting a violation.

Description Optional. Defines how License Scheduler handles distribution policy violations. Distribution policy violations are caused by non-LSF workloads because LSF License Scheduler explicitly follows its distribution policies.

License Scheduler reports a distribution policy violation when the total number of licenses given to the LSF workload, both free and in use, is less than the LSF workload distribution specified in "WORKLOAD_DISTRIBUTION" on page 591. If License Scheduler finds a distribution policy violation, it creates or overwrites the `LSF_LOGDIR/bld.violation.`*`service_domain_name`*`.log` file and runs the user command specified by the `CMD` keyword.

Example The `LicenseServer1` service domain has a total of 80 licenses, and its workload distribution and enforcement is configured as follows:

```
Begin Parameter
...
DISTRIBUTION_POLICY_VIOLATION_ACTION=(PERIOD 5 CMD /bin/mycmd)
...
End Parameter
```

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1(Lp1 1 Lp2 2)
WORKLOAD_DISTRIBUTION=LicenseServer1(LSF 8 NON_LSF 2)
End Feature
```

According to this configuration, 80% of the available licenses, or 64 licenses, are available to the LSF workload. License Scheduler checks the service domain for a violation every five scheduling cycles, and runs the `/bin/mycmd` command if it finds a violation.

If the current LSF workload license usage is 50 and the number of free licenses is 10, the total number of licenses assigned to the LSF workload is 60. This is a violation of the workload distribution policy because this is less than the specified LSF workload distribution of 64 licenses.

ENABLE_INTERACTIVE

Syntax **`ENABLE_INTERACTIVE=Y`**

Description Optional. Globally enables one share of the licenses for interactive tasks.

By default, `ENABLE_INTERACTIVE` is not set. License Scheduler allocates licenses equally to each cluster and does not distribute licenses for interactive tasks.

EXT_FILTER_PORT

Syntax **`EXT_FILTER_PORT=`***integer*

Description Defines the TCP listening port used by all external plugins to communicate with License Scheduler hosts. Specify any non-privileged port number.

It must be the same as the port number specified in EXTERNAL_FILTER_SERVER in the vendor daemon option file. Use a number close to the defined value for the `PORT` parameter. For example, if `PORT`=9581, define `EXT_FILTER_PORT`=9582.

FLX_LICENSE_FILE

Syntax **`FLX_LICENSE_FILE=`***path*

Description Specifies a path to the file that contains the license keys FLEXnet.Ext.Filter and FLEXnet.Usage.Snapshot to enable the FLEXnet APIs.

When bld starts, if LM_LICENSE_FILE environment variable does not exist, it sets the LM_LICENSE_FILE environment variable to FLX_LICENSE_FILE, or appends FLX_LICENSE_FILE to LM_LICENSE_FILE environment variable if it already exists.

HOSTS

Syntax **`HOSTS=`***host_name*`.`*domain_name* ...

Description Defines License Scheduler hosts, including License Scheduler candidate hosts.

Specify a fully qualified host name such as `hostX.mycompany.com`. You can omit the domain name if all your License Scheduler clients run in the same DNS domain.

LIB_RECVTIMEOUT

Syntax **`LIB_RECVTIMEOUT=`***seconds*

Description Specifies a timeout value in seconds for communication between LSF License Scheduler and LSF.

Default 5 seconds

LM_REMOVE_INTERVAL

Syntax **`LM_REMOVE_INTERVAL=`***seconds*

Description Specifies the minimum time a job must have a license checked out before `lmremove` can remove the license. `lmremove` causes `lmgrd` and vendor daemons to close the TCP connection with the application. They will then retry the license checkout.

Default 180 seconds

LM_STAT_INTERVAL

Syntax **`LM_STAT_INTERVAL=`***seconds*

Description Defines a time interval between calls that License Scheduler makes to collect license usage information from FLEXnet license management.

Default 60 seconds

LMSTAT_PATH

Syntax **`LMSTAT_PATH=`***path*

Description Defines the full path to the location of the FLEXnet command `lmstat`.

LS_MAX_TASKMAN_SESSIONS

Syntax **LS_MAX_TASKMAN_SESSIONS=***integer*

Description Defines the maximum number of `taskman` jobs that run simultaneously. This prevents system-wide performance issues that occur if there are a large number of `taskman` jobs running in License Scheduler.

PORT

Syntax **PORT=***integer*

Description Defines the TCP listening port used by License Scheduler hosts, including candidate License Scheduler hosts. Specify any non-privileged port number.

SCHED_INTERVAL

Syntax **SCHED_INTERVAL=***seconds*

Description The regulating time interval in which `bld` performs its regular scheduling operations.

Clusters Section

Description

Required. Lists the clusters that can use License Scheduler.

When configuring clusters for a WAN, the Clusters Section of the master cluster must define its slave clusters.

Clusters section structure

The Clusters section begins and ends with the lines `Begin Clusters` and `End Clusters`. The second line is the column heading, `CLUSTERS`. Subsequent lines list participating clusters, one name per line:

```
Begin Clusters
CLUSTERS
cluster1
cluster2
.
.
End Clusters
```

CLUSTERS

Defines the name of each participating LSF cluster. Specify using one name per line.

ServiceDomain Section

Description

Required. Defines License Scheduler service domains as groups of physical license server hosts that serve a specific network.

ServiceDomain section structure

Define a section for each License Scheduler service domain.

This example shows the structure of the section:

```
Begin ServiceDomain
NAME=DesignCenterB
LIC_SERVERS=((1888@hostD)(1888@hostE))
LIC_COLLECTOR=CenterB
End ServiceDomain
```

Parameters

- "NAME"
- "LIC_SERVERS"
- "LIC_COLLECTOR"
- "LIC_FLEX_API_ENABLE"

NAME

Defines the name of the service domain.

LIC_SERVERS

Syntax **LIC_SERVERS=(**[**(***host_name* | *port_number***@***host_name* |**(***port_number***@***host_name* *port_number***@***host_name* *port_number***@***host_name***))**] ...**)**

Description Defines the FLEXnet license server hosts that make up the License Scheduler service domain. For each FLEXnet license server host, specify the number of the port that FLEXnet uses, then the at symbol (@), then the name of the host. If FLEXnet uses the default port on a host, you can specify the host name without the port number. Put one set of round brackets around the list, and one more set of round brackets around each host, unless you have redundant servers (three hosts sharing one license file). If you have redundant servers, the brackets enclose all three hosts.

Examples

- One FLEXnet license server host:

  `LIC_SERVERS=((1700@hostA))`

- Multiple FLEXnet license server hosts with unique `license.dat` files:

  `LIC_SERVERS=((1700@hostA)(1700@hostB)(1700@hostC))`

- Redundant FLEXnet license server hosts sharing the same `license.dat` file:

  `LIC_SERVERS=((1700@hostD 1700@hostE 1700@hostF))`

LIC_COLLECTOR

Syntax **LIC_COLLECTOR=***licence_collector_name*

Description Optional. Defines a name for the license collector daemon (`blcollect`) to use in each service domain. `blcollect` collects license usage information from FLEXnet and passes it to the License Scheduler daemon (`bld`). It improves performance by allowing you to distribute license information queries on multiple hosts.

You can only specify one collector per service domain, but you can specify one collector to serve multiple service domains. Each time you run `blcollect`, you must specify the name of the collector for the service domain. You can use any name you want.

Default Undefined. The License Scheduler daemon uses one license collector daemon for the entire cluster.

LIC_FLEX_API_ENABLE

Syntax **`LIC_FLEX_API_ENABLE=y | n`**

Description Enables the Macrovision FLEXnet APIs to replace the default behaviour of scheduling based on `lmstat` data.

You must also configure License Scheduler and your vendor daemons to work with the Macrovision FLEXnet Manager package.

Default `N` (License Scheduler uses `lmstat` for scheduling)

Feature Section

Description

Required. Defines license distribution policies.

Feature section structure

Define a section for each feature managed by License Scheduler.

```
Begin Feature
NAME=vcs
FLEX_NAME=vcs
DISTRIBUTION=lanserver1 (Lp1 1 Lp2 4/6)
lanserver2 (Lp3 1 Lp4 10/8)
wanserver (Lp1 1 Lp2 1 Lp3 1 Lp4 1)
End Feature
```

Parameters

- "NAME"
- "FLEX_NAME"
- "DISTRIBUTION"
- "ALLOCATION"
- "GROUP"
- "GROUP_DISTRIBUTION"
- "NON_SHARED_DISTRIBUTION"
- "PREEMPT_RESERVE"
- "SERVICE_DOMAINS"
- "WORKLOAD_DISTRIBUTION"
- "ENABLE_DYNAMIC_RUSAGE"
- "DYNAMIC"

NAME

Required. Defines the token name—the name used by License Scheduler and LSF to identify the license feature. Normally, the token name should be the same as the FLEXnet feature name, as they represent the same license.

Normally, license token names should be the same as the FLEXnet Licensing feature names, as they represent the same license. However, LSF does not support names that start with a number, or names containing a dash or hyphen character (-), which may be used in the FLEXnet Licensing feature name.

FLEX_NAME

Optional. Defines the feature name—the name used by FLEXnet to identify the type of license. You only need to specify this parameter if the License Scheduler token name is not identical to the FLEXnet feature name.

FLEX_NAME allows the NAME parameter to be an alias of the FLEXnet feature name. For feature names that start with a number or contain a dash (-), you must set both NAME and FLEX_NAME, where FLEX_NAME is the actual FLEXnet Licensing feature name, and NAME is an arbitrary license token name you choose.

For example

```
Begin Feature
FLEX_NAME=201-AppZ
NAME=AppZ201
DISTRIBUTION=LanServer1(Lp1 1 Lp2 1)
End Feature
```

DISTRIBUTION

Syntax

DISTRIBUTION=[*service_domain_name* **(**[*project_name* *number_shares*[/*number_licenses_owned*]] ... [**default**] **)**] ...

- *service_domain_name*

  Specify a License Scheduler service domain (described in the "ServiceDomain Section" on page 583) that distributes the licenses.

- *project_name*

  Specify a License Scheduler project (described in the "Projects Section" on page 596) that is allowed to use the licenses.

- *number_shares*

  Specify a positive integer representing the number of shares assigned to the project.

  The number of shares assigned to a project is only meaningful when you compare it to the number assigned to other projects, or to the total number assigned by the service domain. The total number of shares is the sum of the shares assigned to each project.

- *number_licenses_owned*

  Optional. Specify a slash (/) and a positive integer representing the number of licenses that the project owns.

- `default`

  A reserved keyword that represents the default License Scheduler project if the job submission does not specify a project (`bsub -Lp`).

Description

Required if GROUP_DISTRIBUTION is not defined. Defines the distribution policies for the license. The name of each service domain is followed by its distribution policy, in parentheses. The distribution policy determines how the licenses available in each service domain are distributed among the clients.

The distribution policy is a space-separated list with each project name followed by its share assignment. The share assignment determines what fraction of available licenses is assigned to each project, in the event of competition between projects. Optionally, the share assignment is followed by a slash and the number of licenses owned by that project. License ownership enables a preemption policy. (In the event of competition between projects, projects that own licenses preempt jobs. Licenses are returned to the owner immediately.)

GROUP_DISTRIBUTION and DISTRIBUTION are mutually exclusive. If they are both defined in the same feature, the License Scheduler daemon returns an error and ignores this feature.

Examples

`DISTRIBUTION=wanserver (Lp1 1 Lp2 1 Lp3 1 Lp4 1)`

In this example, the service domain named `wanserver` shares licenses equally among four License Scheduler projects. If all projects are competing for a total of eight licenses, each project is entitled to two licenses at all times. If all projects are competing for only two licenses in total, each project is entitled to a license half the time.

`DISTRIBUTION=lanserver1 (Lp1 1 Lp2 2/6)`

In this example, the service domain named `lanserver1` allows `Lp1` to use one third of the available licenses and `Lp2` can use two thirds of the licenses. However, `Lp2` is always entitled to six licenses, and can preempt another project to get the licenses immediately if they are needed. If the projects are competing for a total of 12 licenses, `Lp2` is entitled to eight licenses (six on demand, and two more as soon as they are free). If the projects are competing for only six licenses in total, `Lp2` is entitled to all of them, and `Lp1` can only use licenses when `Lp2` does not need them.

ALLOCATION

Syntax

ALLOCATION=*project_name* **(***cluster_name* [*number_shares*] ... **)**] ...

- *cluster_name*

  Specify LSF cluster names that licenses are to be allocated to.
- *project_name*

  Specify a License Scheduler project (described in the PROJECTS section) that is allowed to use the licenses.
- *number_shares*

  Specify a positive integer representing the number of shares assigned to the cluster.

  The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters. The total number of shares is the sum of the shares assigned to each cluster.

Description

Defines the allocation of license features across clusters and between LSF jobs and non-LSF interactive jobs.

ALLOCATION ignores the global setting of the `ENABLE_INTERACTIVE` parameter because `ALLOCATION` is configured for the license feature.

You can configure the allocation of license shares to:

- Change the share number between clusters for a feature
- Limit the scope of license usage and change the share number between LSF jobs and interactive tasks for a feature

To manage interactive (non-LSF) tasks in License Scheduler projects, you require the LSF Task Manager, `taskman`. The Task Manager utility is supported by but not shipped with License Scheduler. For more information about `taskman`, contact Platform.

Default Undefined. If ENABLE_INTERACTIVE is not set, each cluster receives one share, and interactive tasks receive no shares.

Each example contains two clusters and 12 licenses of a specific feature.

Example 1 ALLOCATION is not configured. The ENABLE_INTERACTIVE parameter is not set.

```
Begin Parameters
...
ENABLE_INTERACTIVE=n
...
End Parameters
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1 (Lp1 1)
End Feature
```

Six licenses are allocated to each cluster. No licenses are allocated to interactive tasks.

Example 2 ALLOCATION is not configured. The ENABLE_INTERACTIVE parameter is set.

```
Begin Parameters
...
ENABLE_INTERACTIVE=y
...
End Parameters
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1 (Lp1 1)
End Feature
```

Four licenses are allocated to each cluster. Four licenses are allocated to interactive tasks.

Example 3 In the following example, the ENABLE_INTERACTIVE parameter does not affect the ALLOCATION configuration of the feature.

ALLOCATION is configured. The ENABLE_INTERACTIVE parameter is set.

```
Begin Parameters
...
ENABLE_INTERACTIVE=y
...
End Parameters
Begin Feature
NAME=ApplicationY
DISTRIBUTION=LicenseServer1 (Lp2 1)
ALLOCATION=Lp2(cluster1 1 cluster2 0 interactive 1)
End Feature
```

The ENABLE_INTERACTIVE setting is ignored. Licenses are shared equally between cluster1 and interactive tasks. Six licenses of ApplicationY are allocated to cluster1. Six licenses are allocated to interactive tasks.

Example 4 In the following example, the ENABLE_INTERACTIVE parameter does not affect the ALLOCATION configuration of the feature.

ALLOCATION is configured. The ENABLE_INTERACTIVE parameter is not set.

```
Begin Parameters
...
ENABLE_INTERACTIVE=n
...
End Parameters
Begin Feature
NAME=ApplicationZ
DISTRIBUTION=LicenseServer1 (Lp1 1)
ALLOCATION=Lp1(cluster1 0 cluster2 1 interactive 2)
End Feature
```

The `ENABLE_INTERACTIVE` setting is ignored. Four licenses of `ApplicationZ` are allocated to `cluster2`. Eight licenses are allocated to interactive tasks.

GROUP

Syntax

GROUP=[*group_name* **(***project_name*... **)**] ...

- *group_name*

  Specify a name for a group of projects.

- *project_name*

  Specify a License Scheduler project (described in the PROJECTS section) that is allowed to use the licenses. The project must appear in the DISTRIBUTION.

  A project should only belong to one group.

Description

Optional. Defines groups of projects and specifies the name of each group. The groups defined here are used for group preemption and replace single projects with group projects.

This parameter is ignored if GROUP_DISTRIBUTION is also defined.

GROUP_DISTRIBUTION

Syntax

GROUP_DISTRIBUTION=*top_level_hierarchy_name*

- *top_level_hierarchy_name*

  Specify the name of the top level hierarchical group.

Description

Required if DISTRIBUTION is not defined. Defines the name of the hierarchical group containing the distribution policy attached to this feature.

GROUP_DISTRIBUTION and DISTRIBUTION are mutually exclusive. If they are both defined in the same feature, the License Scheduler daemon returns an error and ignores this feature.

If GROUP is also defined, it is ignored in favour of GROUP_DISTRIBUTION.

Example

In the following example, the GROUP_DISTRIBUTION parameter hierarchical scheduling for the top-level hierarchical group named `groups`. The SERVICE_DOMAINS parameter defines a list of service domains that provide tokens for the group.

```
Begin Feature
NAME = myjob2
GROUP_DISTRIBUTION = groups
SERVICE_DOMAINS = LanServer  wanServer
End Feature
```

NON_SHARED_DISTRIBUTION

Syntax

NON_SHARED_DISTRIBUTION=*service_domain_name* **(**[*project_name number_non_shared_licenses*] ... **)** ...

- *service_domain_name*

  Specify a License Scheduler service domain (described in the "ServiceDomain Section" on page 583) that distributes the licenses.
- *project_name*

  Specify a License Scheduler project (described in the "Projects Section" on page 596) that is allowed to use the licenses.
- *number_non_shared_licenses*

  Specify a positive integer representing the number of non-shared licenses that the project owns.

Description

Optional. Defines non-shared licenses. Non-shared licenses are not shared with other license projects. They are available only to that project.

Use `blinfo -a` to display NON_SHARED_DISTRIBUTION information.

Example

```
Begin Feature
NAME=f1  # total 15 on LanServer and 15 on WanServer
FLEX_NAME=VCS-RUNTIME
DISTRIBUTION=LanServer(Lp1 4 Lp2 1) WanServer (Lp1 1 Lp2 1/3)
NON_SHARED_DISTRIBUTION=LanServer(Lp1 10) WanServer (Lp1 5 Lp2
3)
PREEMPT_RESERVE=Y
End Feature
```

In this example:

- 10 non-shared licenses are defined for the `Lp1` project on LanServer
- 5 non-shared licenses are defined for the `Lp1` project on WanServer
- 3 non-shared licenses are defined for the `Lp2` project on `WanServer`

The remaining licenses are distributed as follows:

- LanServer: The remaining 5 (15-10=5) licenses on `LanServer` will be distributed to the `Lp1` and `Lp2` projects with a 4:1 ratio.
- WanServer: The remaining 7(15-5-3=7) licenses on `WanServer` will be distributed to the `Lp1` and `Lp2` projects with a 1:1 ratio.

If `Lp2` uses fewer than 6 (3 privately owned+ 3 owned) licenses, then a job in the `Lp2` can preempt `Lp1` jobs.

PREEMPT_LSF

Syntax

PREEMPT_LSF=Y

Description Optional. With the Macrovision FLEXnet plugin integration installed, enables on-demand preemption of LSF jobs for important non-managed workload. This guarantees that important non-managed jobs will not fail because of lack of licenses.

Default LSF workload is not preemtable

PREEMPT_RESERVE

Syntax `PREEMPT_RESERVE=Y`

Description Optional. Enables License Scheduler to preempt either licenses that are reserved or already in use by other projects. The number of jobs must be greater than the number of licenses owned.

Default Reserved licenses are not preemtable

SERVICE_DOMAINS

Syntax **SERVICE_DOMAINS=***service_domain_name* ...

- *service_domain_name*

  Specify the name of the service domain.

Description Required if GROUP_DISTRIBUTION is defined. Specifies the service domains that provide tokens for this feature.

WORKLOAD_DISTRIBUTION

Syntax **WORKLOAD_DISTRIBUTION=**[*service_domain_name* **(LSF** *lsf_distribution* [**/***enforced_distribution*] **NON_LSF** *non_lsf_distribution***)**] ...

- *service_domain_name*

  Specify a License Scheduler service domain (described in the "ServiceDomain Section" on page 583) that distributes the licenses.

- *lsf_distribution*

  Specify the share of licenses dedicated to LSF workloads. The share of licenses dedicated to LSF workloads is a ratio of *lsf_distribution*:*non_lsf_distribution*.

- *enforced_distribution*

  Optional. Specify a slash (`/`) and a positive integer representing the enforced number of licenses. License Scheduler will not reserve more than this number of licenses for LSF workloads regardless of the specified value of *lsf_distribution*.

- *non_lsf_distribution*

  Specify the share of licenses dedicated to non-LSF workloads. The share of licenses dedicated to non-LSF workloads is a ratio of *non_lsf_distribution*:*lsf_distribution*.

Description Optional. Defines the distribution given to each LSF and non-LSF workload within the specified service domain.

Use `blinfo -a` to display WORKLOAD_DISTRIBUTION configuration.

Example 1

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1(Lp1 1 Lp2 2)
WORKLOAD_DISTRIBUTION=LicenseServer1(LSF 8 NON_LSF 2)
End Feature
```

On the `LicenseServer1` domain, the available licenses are dedicated in a ratio of 8:2 for LSF and non-LSF workloads. This means that 80% of the available licenses are dedicated to the LSF workload, and 20% of the available licenses are dedicated to the non-LSF workload.

If `LicenseServer1` has a total of 80 licenses, this configuration indicates that 64 licenses are dedicated to the LSF workload, and 16 licenses are dedicated to the non-LSF workload.

Example 2

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1(Lp1 1 Lp2 2)
WORKLOAD_DISTRIBUTION=LicenseServer1(LSF 8/40 NON_LSF 2)
End Feature
```

On the `LicenseServer1` domain, the available licenses are dedicated in a ratio of 8:2 for LSF and non-LSF workloads, with an absolute maximum of 40 licenses dedicated to the LSF workload. This means that 80% of the available licenses, up to a maximum of 40, are dedicated to the LSF workload, and the remaining licenses are dedicated to the non-LSF workload.

If `LicenseServer1` has a total of 40 licenses, this configuration indicates that 32 licenses are dedicated to the LSF workload, and eight licenses are dedicated to the non-LSF workload. However, if `LicenseServer1` has a total of 80 licenses, only 40 licenses are dedicated to the LSF workload, and the remaining 40 licenses are dedicated to the non-LSF workload.

ENABLE_DYNAMIC_RUSAGE

Syntax **ENABLE_DYNAMIC_RUSAGE=Y**

Description

Enforces license distribution policies for class C liense features.

When set, ENABLE_DYNAMIC_RUSAGE enables all class-C license checkouts to be considered managed checkout, instead of unmanaged (or OTHERS).

DYNAMIC

Syntax **DYNAMIC=Y**

Description

If you specify DYNAMIC=Y, you must specify a duration in an rusage resource requirement for the feature. This enables License Scheduler to treat the license as a dynamic resource and prevents License Scheduler from scheduling tokens for the feature when they are not available, or reserving license tokesn when they should actually be free.

ProjectGroup Section

Description

Optional. Defines the hierarchical relationships of projects.

The hierarchical groups that can have multiple levels of grouping. You can configure a tree-like scheduling policy, with the leaves being the license projects that jobs can belong to. Each project group in the tree has a set of values, including shares, limits, ownership and non-shared, or exclusive, licenses.

Use `blstat -G` to view the hierarchical dynamic license information.

Use `blinfo -G` to view the hierarchical configuration.

Feature section structure

Define a section for each hierarchical group managed by License Scheduler.

The keywords GROUP, SHARES, OWNERSHIP, LIMIT, and NON_SHARED are required, empty brackets are allowed only for OWNERSHIP and LIMIT. SHARES must be specified.

```
Begin ProjectGroup
GROUP             SHARES     OWNERSHIP    LIMITS      NON_SHARED
(licgrp1 (a b))   (1 1)      ()           (10 10)     (4 4)
(a (c d))         (1 1)      ()           (10 10)     (0 4)
(b (e f))         (1 1)      ()           (- 5)       (2 2)
(c (1 2 3))       (1 1 2)    ()           (3 4 5)     ()
(d (4 5 6))       (1 1 1)    (1 1 1)      ()          (- 3 0)
(e (7 8 9))       (1 1 1)    (2 - 2)      ()          (1 - 1)
(f (10 11 12))    (1 1 1)    (2 2 2)      (4 4 4)     (1 0 1)
End ProjectGroup
```

Parameters

- “GROUP”
- “SHARES”
- “OWNERSHIP”
- “LIMITS”
- “NON_SHARED”

GROUP

Defines the project names in the hierarchical grouping and its relationships. Each entry specifies the name of the hierarchical group and its members.

For better readability, you should specify the projects in the order from the root to the leaves as in example.

Specify the entry as follows:

(*group* **(***member ...***))**

SHARES

Required. Defines the shares assigned to the hierarchical group member projects. Specify the share for each member, separated by spaces, in the same order as listed in the GROUP column.

OWNERSHIP

Defines the level of ownership of the hierarchical group member projects. Specify the ownership for each member, separated by spaces, in the same order as listed in the GROUP column.

You can only define OWNERSHIP for hierarchical group member projects, not hierarchical groups. Do not define a number for the top level project group.

A dash (-) is equivalent to a zero, which means there are no owners of the projects. You can leave the parentheses empty `()` if desired.

Valid values

A positive integer between the NON_SHARED and LIMITS values defined for the specified hierarchical group.

- If defined as less than NON_SHARED, OWNERSHIP is set to NON_SHARED.
- If defined as greater than LIMITS, OWNERSHIP is set to LIMITS.

LIMITS

Defines the maximum number of licenses that can be used at any one time by the hierarchical group member projects. Specify the maximum number of licenses for each member, separated by spaces, in the same order as listed in the GROUP column.

A dash (-) is equivalent to `INFINIT_INT`, which means there is no maximum limit and the project group can use as many licenses as possible.

You can leave the parentheses empty `()` if desired.

NON_SHARED

Defines the number of licenses that the hierarchical group member projects use exclusively. Specify the number of licenses for each group or project, separated by spaces, in the same order as listed in the GROUP column.

A dash (-) is equivalent to a zero, which means there are no licenses that the hierarchical group member projects use exclusively.

Normally, the total number of non-shared licenses should be less than the total number of license tokens available. License tokens may not be available to project groups if the total non-shared licenses for all groups is greater than the number of shared tokens available.

For example, feature `p4_4` is configured as follows, with a total of 4 tokens:

```
Begin Feature
NAME =p4_4 # total token value is 4
GROUP_DISTRIBUTION=final
SERVICE_DOMAINS=LanServer
End Feature
```

To correct configuration is:

```
GROUP              SHARES   OWNERSHIP   LIMITS     NON_SHARED
(final (G2 G1))    (1 1)    ()          ()         (2 0)
(G1 (AP2 AP1))     (1 1)    ()          ()         (1 1)
```

Valid values Any positive integer up to the LIMITS value defined for the specified hierarchical group.

If defined as greater than LIMITS, NON_SHARED is set to LIMITS.

Projects Section

Description

Required. Lists the License Scheduler projects.

Projects section structure

The Projects section begins and ends with the lines `Begin Projects` and `End Projects`. The second line consists of the required column heading `PROJECTS` and the optional column heading `PRIORITY`. Subsequent lines list participating projects, one name per line.

Examples

The following example lists the projects without defining the priority:

```
Begin Projects
PROJECTS
Lp1
Lp2
Lp3
Lp4
.
.
End Projects
```

The following example lists the projects and defines the priority of each project:

```
Begin Projects
PROJECTS          PRIORITY
Lp1                3
Lp2                4
Lp3                2
Lp4                1
default            0
.
.
End Projects
```

Parameters

- "PROJECTS"
- "PRIORITY"

PROJECTS

Defines the name of each participating project. Specify using one name per line.

PRIORITY

Optional. Defines the priority for each project where "0" is the lowest priority, and the higher number specifies a higher priority. This column overrides the default behavior. Instead of preempting in order the projects are listed under PROJECTS based on the accumulative `inuse` usage of each project, the projects are preempted according to the specified priority from lowest to highest., When 2 projects have the same priority number configured, the first listed project has higher priority, like LSF queues.

Priority of default project

If not explicitly configured, the default project has the priority of 0. You can override this value by explicitly configuring the default project in Projects section with the chosen priority value.

SEE ALSO

blcollect(1), bladmin(8), lsf.conf(5)

lsf.shared

The `lsf.shared` file contains common definitions that are shared by all load sharing clusters defined by `lsf.cluster.`*cluster_name* files. This includes lists of cluster names, host types, host models, the special resources available, and external load indices.

This file is installed by default in the directory defined by LSF_CONFDIR.

Changing lsf.shared configuration

After making any changes to `lsf.shared`, run the following commands:

- `lsadmin reconfig` to reconfigure LIM
- `badmin mbdrestart` to restart `mbatchd`

Contents

- "Cluster Section" on page 600
- "HostType Section" on page 601
- "HostModel Section" on page 602
- "Resource Section" on page 604

Cluster Section

(Required) Lists the cluster names recognized by the LSF system

Cluster section structure

The first line must contain the mandatory keyword ClusterName. The other keyword is optional.

The first line must contain the mandatory keyword ClusterName and the keyword Servers in a MultiCluster environment.

Each subsequent line defines one cluster.

Example Cluster section

```
Begin Cluster
ClusterName  Servers
cluster1     hostA
cluster2     hostB
End Cluster
```

ClusterName

Defines all cluster names recognized by the LSF system.

All cluster names referenced anywhere in the LSF system must be defined here. The file names of cluster-specific configuration files must end with the associated cluster name.

By default, if MultiCluster is installed, all clusters listed in this section participate in the same MultiCluster environment. However, individual clusters can restrict their MultiCluster participation by specifying a subset of clusters at the cluster level (`lsf.cluster.`*cluster_name* RemoteClusters section).

Servers

MultiCluster only. List of hosts in this cluster that LIMs in remote clusters can connect to and obtain information from.

For other clusters to work with this cluster, one of these hosts must be running `mbatchd`.

HostType Section

(Required) Lists the valid host types in the cluster. All hosts that can run the same binary executable are in the same host type.

HostType section structure

The first line consists of the mandatory keyword TYPENAME.

Subsequent lines name valid host types.

Example HostType section

```
Begin HostType
TYPENAME
SUN41
SOLSPARC
ALPHA
HPPA
NTX86
End HostType
```

TYPENAME

Host type names are usually based on a combination of the hardware name and operating system. If your site already has a system for naming host types, you can use the same names for LSF.

HostModel Section

(Required) Lists models of machines and gives the relative CPU scaling factor for each model. All hosts of the same relative speed are assigned the same host model.

LSF uses the relative CPU scaling factor to normalize the CPU load indices so that jobs are more likely to be sent to faster hosts. The CPU factor affects the calculation of job execution time limits and accounting. Using large or inaccurate values for the CPU factor can cause confusing results when CPU time limits or accounting are used.

HostModel section structure

The first line consists of the mandatory keywords MODELNAME, CPUFACTOR, and ARCHITECTURE.

Subsequent lines define a model and its CPU factor.

Example HostModel section

```
Begin HostModel
MODELNAME  CPUFACTOR     ARCHITECTURE
PC400        13.0        (i86pc_400 i686_400)
PC450        13.2        (i86pc_450 i686_450)
Sparc5F       3.0        (SUNWSPARCstation5_170_sparc)
Sparc20       4.7        (SUNWSPARCstation20_151_sparc)
Ultra5S      10.3        (SUNWUltra5_270_sparcv9 SUNWUltra510_270_sparcv9)
End HostModel
```

ARCHITECTURE

Description (Reserved for system use only) Indicates automatically detected host models that correspond to the model names.

CPUFACTOR

Description Though it is not required, you would typically assign a CPU factor of 1.0 to the slowest machine model in your system and higher numbers for the others. For example, for a machine model that executes at twice the speed of your slowest model, a factor of 2.0 should be assigned.

MODELNAME

Description Generally, you need to identify the distinct host types in your system, such as MIPS and SPARC first, and then the machine models within each, such as SparcIPC, Sparc1, Sparc2, and Sparc10.

About automatically detected host models and types

When you first install LSF, you do not necessarily need to assign models and types to hosts in `lsf.cluster.`*cluster_name*. If you do not assign models and types to hosts in `lsf.cluster.`*cluster_name*, LIM automatically detects the model and type for the host.

If you have versions earlier than LSF 4.0, you may have host models and types already assigned to hosts. You can take advantage of automatic detection of host model and type also.

Automatic detection of host model and type is useful because you no longer need to make changes in the configuration files when you upgrade the operating system or hardware of a host and reconfigure the cluster. LSF will automatically detect the change.

Mapping to CPU factors

Automatically detected models are mapped to the short model names in `lsf.shared` in the ARCHITECTURE column. Model strings in the ARCHITECTURE column are only used for mapping to the short model names.

Example `lsf.shared` file:

```
Begin HostModel
MODELNAME   CPUFACTOR     ARCHITECTURE
SparcU5     5.0           (SUNWUltra510_270_sparcv9)
PC486       2.0           (i486_33 i486_66)
PowerPC     3.0           (PowerPC12 PowerPC16 PowerPC31)
End HostModel
```

If an automatically detected host model cannot be matched with the short model name, it is matched to the best partial match and a warning message is generated.

If a host model cannot be detected or is not supported, it is assigned the DEFAULT model name and an error message is generated.

Naming convention

Models that are automatically detected are named according to the following convention:

hardware_platform [*_processor_speed*[*_processor_type*]]

where:

- *hardware_platform* is the only mandatory component
- *processor_speed* is the optional clock speed and is used to differentiate computers within a single platform
- *processor_type* is the optional processor manufacturer used to differentiate processors with the same speed
- Underscores (_) between *hardware_platform*, *processor_speed*, *processor_type* are mandatory.

Resource Section

Optional. Defines resources (must be done by the LSF administrator).

Resource section structure

The first line consists of the keywords. RESOURCENAME and DESCRIPTION are mandatory. The other keywords are optional. Subsequent lines define resources.

Example Resource section

```
Begin Resource
RESOURCENAME     TYPE      INTERVAL    INCREASING   RELEASE   DESCRIPTION
mips             Boolean      ()           ()          ()     (MIPS architecture)
dec              Boolean      ()           ()          ()     (DECStation system)
sparc            Boolean      ()           ()          ()     (SUN SPARC)
bsd              Boolean      ()           ()          ()     (BSD unix)
hpux             Boolean      ()           ()          ()     (HP-UX UNIX)
aix              Boolean      ()           ()          ()     (AIX UNIX)
solaris          Boolean      ()           ()          ()     (SUN SOLARIS)
bigmem           Boolean      ()           ()          ()     (host with big memory)
myResource       String       ()           ()          ()     (MIPS architecture)
static_sh1       Numeric      ()           N           ()     (static)
external_1       Numeric      15           Y           ()     (external)
End Resource
```

RESOURCENAME

Description The name you assign to the new resource. An arbitrary character string.

- A resource name cannot begin with a number.
- A resource name cannot contain any of the following characters:

  `:  .  (  )  [  +  -  *  /  !  &  |  <  >  @  =`
- A resource name cannot be any of the following reserved names:

  `cpu cpuf io logins ls idle maxmem maxswp maxtmp type model status it mem ncpus ndisks pg r15m r15s r1m swap swp tmp ut`
- Resource names are case sensitive
- Resource names can be up to 29 characters in length

TYPE

Description The type of resource:

- Boolean—Resources that have a value of 1 on hosts that have the resource and 0 otherwise.
- Numeric—Resources that take numerical values, such as all the load indices, number of processors on a host, or host CPU factor.
- String— Resources that take string values, such as host type, host model, host status.

Default If TYPE is not given, the default type is Boolean.

DESCRIPTION

Description Brief description of the resource.

The information defined here will be returned by the ls_info() API call or printed out by the lsinfo command as an explanation of the meaning of the resource.

INCREASING

Applies to numeric resources only.

Description

If a larger value means greater load, INCREASING should be defined as Y. If a smaller value means greater load, INCREASING should be defined as N.

INTERVAL

Optional. Applies to dynamic resources only.

Description

Defines the time interval (in seconds) at which the resource is sampled by the ELIM.

If INTERVAL is defined for a numeric resource, it becomes an external load index.

Default

If INTERVAL is not given, the resource is considered static.

RELEASE

Applies to numeric shared resources only, such as floating licenses.

Description

Controls 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 this parameter.

Specify N to hold the resource, or specify Y to release the resource.

Default

Y

lsf.sudoers

Contents

- "About lsf.sudoers" on page 608
- "lsf.sudoers on UNIX" on page 609
- "lsf.sudoers on Windows" on page 610
- "File Format" on page 611
- "Creating and Modifying lsf.sudoers" on page 612
- "Parameters" on page 613

About lsf.sudoers

The `lsf.sudoers` file is an optional file to configure security mechanisms. It is not installed by default.

You use `lsf.sudoers` to set the parameter LSF_EAUTH_KEY to configure a key for `eauth` to encrypt and decrypt user authentication data.

On UNIX, you also use `lsf.sudoers` to grant permission to users other than root to perform certain operations as root in LSF, or as a specified user.

These operations include:

- LSF daemon startup/shutdown
- User ID for LSF authentication
- User ID for LSF pre- and post-execution commands.
- User ID for external LSF executables

If `lsf.sudoers` does not exist, only root can perform these operations in LSF on UNIX.

On UNIX, this file is located in `/etc`.

There is one `lsf.sudoers` file per host.

On Windows, this file is located in the directory specified by the parameter LSF_SECUREDIR in `lsf.conf`.

Changing lsf.sudoers configuration

After making any changes to `lsf.sudoers`, run `badmin reconfig` to reload the configuration files.

lsf.sudoers on UNIX

In LSF, certain operations such as daemon startup can only be performed by root. The `lsf.sudoers` file grants root privileges to specific users or user groups to perform these operations.

Location

`lsf.sudoers` must be located in `/etc` on each host.

Permissions

`lsf.sudoers` must have permission 600 and be readable and writable only by root.

lsf.sudoers on Windows

Location

The `lsf.sudoers` file is shared over an NTFS network, not duplicated on every Windows host.

By default, LSF installs `lsf.sudoers` in the %SYSTEMROOT% directory.

The location of `lsf.sudoers` on Windows must be specified by LSF_SECUREDIR in `lsf.conf`. You must configure the LSF_SECUREDIR parameter in `lsf.conf` if using `lsf.sudoers` on Windows.

Permissions

The permissions on `lsf.sudoers` for Windows are:

Workgroup Environment

- Local Admins (W)
- Everyone (R)

Domain Environment

- Domain Admins (W)
- Everyone (R)

File Format

The format of `lsf.sudoers` is very similar to that of `lsf.conf`.

Each entry can have one of the following forms:

- `NAME=VALUE`
- `NAME=`
- `NAME= "STRING1 STRING2 ..."`

The equal sign `=` must follow each `NAME` even if no value follows and there should be no space beside the equal sign.

`NAME` describes an authorized operation.

`VALUE` is a single string or multiple strings separated by spaces and enclosed in quotation marks.

Lines starting with a pound sign (#) are comments and are ignored. Do not use `#if` as this is reserved syntax for time-based configuration.

Example lsf.sudoers File

```
LSB_PRE_POST_EXEC_USER=user100
LSF_STARTUP_PATH=/usr/share/lsf/etc
LSF_STARTUP_USERS="user1 user10 user55"
```

Creating and Modifying lsf.sudoers

You can create and modify `lsf.sudoers` with a text editor such as `vi`.

On Windows, you can use the graphical tool `xlsadmin` to create or modify `lsf.sudoers`, by selecting **Configure** | **Security Parameters**. You must invoke `xlsadmin` as a domain administrator for a Windows domain. For a Windows workgroup, you must invoke `xlsadmin` as a local user with the necessary administrative privileges.

After you modify `lsf.sudoers`, you need to restart all `sbatchd`s in the cluster with the command `badmin hrestart all` to update configuration.

Parameters

- "LSB_PRE_POST_EXEC_USER"
- "LSF_EAUTH_KEY"
- "LSF_EAUTH_USER"
- "LSF_EEXEC_USER"
- "LSF_LOAD_PLUGINS"
- "LSF_STARTUP_USERS"
- "LSF_STARTUP_PATH"

LSB_PRE_POST_EXEC_USER

Syntax **LSB_PRE_POST_EXEC_USER**=*user_name*

Description UNIX only.

Specifies the authorized user for running queue level pre-execution and post-execution commands. When this parameter is defined, the queue level pre-execution and post-execution commands will be run as the specified user.

In particular, you can define this parameter if you need to run commands as root on UNIX.

If you configure this parameter as root, the `LD_PRELOAD` and `LD_LIBRARY_PATH` variables are removed from the pre-execution, post-execution, and eexec environments for security purposes.

Pre-execution and post-execution commands are configured at the queue level by the LSF administrator.

You can only define a single user name in this parameter.

Default Undefined. Pre-execution and post-execution commands are run as the user who submitted the job.

LSF_EAUTH_KEY

Syntax **LSF_EAUTH_KEY**=*key*

Description UNIX and Windows.

Specifies a key `eauth` uses to encrypt and decrypt user authentication data.

This parameter provides a way to increase security at a site. The rule to choosing a key is the same as for choosing a password.

If you want to improve the security of your site by specifying a key, make sure it is at least six characters long and uses only printable characters (as when choosing a normal UNIX password).

If you want to change the key, modify the `lsf.sudoers` file on every host. For the hosts to work together, they must all use the same key.

Default Undefined. `eauth` encrypts and decrypts authentication data using an internal key.

LSF_EAUTH_USER

Syntax **LSF_EAUTH_USER**=*user_name*

Description UNIX only.

Specifies the user account under which to run the external authentication executable `eauth`.

Default Undefined. `eauth` is run as the primary LSF administrator.

LSF_EEXEC_USER

Syntax **`LSF_EEXEC_USER`**=*user_name*

Description UNIX only.

Defines the user name to run the external execution command `eexec`.

Default Undefined. `eexec` is run as the user who submitted the job.

LSF_LOAD_PLUGINS

Syntax **`LSF_LOAD_PLUGINS=y`** | **`Y`**

Description If defined, LSF loads plugins from LSB_LSBDIR. Used for Kerberos authentication in Sun HPC environments, and to enable the LSF cpuset plugin for IRIX 6.5.8.

Default Undefined (no plugins).

LSF_STARTUP_USERS

Syntax **`LSF_STARTUP_USERS`**=**`all_admins`** | **"***user_name*...**"**

Description UNIX only. Equivalent to the local LSF administrators group (Local Admins) in Windows.

Must be defined in conjunction with LSF_STARTUP_PATH for this feature to work.

By default, only root can start the LSF daemons. `lsadmin` and `badmin` must be installed as `setuid root` programs.

This parameter specifies other users who can start daemons as root using the LSF administration commands `lsadmin` and `badmin`.

- **`all_admins`**

  Allows all LSF administrators configured in `lsf.cluster.`*cluster_name* to start LSF daemons as root by running `lsadmin` and `badmin` commands.

  Defining LSF_STARTUP_USERS as `all_admins` incurs some security risk because administrators can be configured by a primary LSF administrator who is not root. You should explicitly list the login names of all authorized administrators here so that you have full control of who can start daemons as root.

- **"***user_name*...**"**

  Allows specified users to start LSF daemons as root by running `lsadmin` and `badmin` commands. If only one user is specified, quotation marks are not required.

Default Undefined. Only root can start daemons as root.

See Also LSF_STARTUP_PATH

LSF_STARTUP_PATH

Syntax **LSF_STARTUP_PATH=***path*

Description UNIX only.

Absolute path name of the directory in which the server binaries (LIM, RES, `sbatchd`, `mbatchd`, etc.) are installed.

This is normally LSF_SERVERDIR as defined in `cshrc.lsf`, `profile.lsf` or `lsf.conf`. LSF will allow the specified administrators (see “LSF_STARTUP_USERS” on page 614) to start the daemons installed in the LSF_STARTUP_PATH directory.

Both LSF_STARTUP_USERS and LSF_STARTUP_PATH must be defined for this feature to work.

Default Undefined

See Also LSF_STARTUP_USERS

SEE ALSO

`lsadmin`(8), `badmin`(8), `lsf.conf`(5), `lsfstartup`(3), `lsf.cluster`(5), `eexec`(8), `eauth`(8)

.

lsf.task

Users should not have to specify a resource requirement each time they submit a job. LSF supports the concept of a task list. This chapter describes the files used to configure task lists:

- `lsf.task`
- `lsf.task.`*`cluster_name`*
- `.lsftask`

Changing task list configuration

After making any changes to the task list files, run the following commands:

- `lsadmin reconfig` to reconfigure LIM
- `badmin reconfig` to reload the configuration files

Contents

- "About Task Lists" on page 618
- "Task Files" on page 619
- "Format of Task Files" on page 620

About Task Lists

A task list is a list in LSF that keeps track of the default resource requirements for different applications and task eligibility for remote execution.

The term task refers to an application name. With a task list defined, LSF automatically supplies the resource requirement of the job whenever users submit a job unless one is explicitly specified at job submission.

LSF takes the job's command name as the task name and uses that name to find the matching resource requirement for the job from the task list. If a task does not have an entry in the task list, LSF assumes the default resource requirement; that is, a host that has the same host type as the submission host will be chosen to run the job.

An application listed in a task file is considered for load sharing by its placement in either the local tasks or remote tasks list.

- A local task is typically an application or command that it does not make sense to run remotely such as `ls`.
- A remote task is an application or command that can be run on another machine in the LSF cluster. The `compress` command is an example of a remote task.

Some applications require resources other than the default. LSF can store resource requirements for specific applications in remote task list files, so that LSF automatically chooses candidate hosts that have the correct resources available.

For frequently used commands and software packages, the LSF administrator can set up cluster–wide resource requirements that apply to all users in the cluster.

Users can modify and add to these requirements by setting up additional resource requirements that apply only to their own jobs.

Cluster-wide resource requirements

The resource requirements of applications are stored in the remote task list file.

LSF automatically picks up a job's default resource requirement string from the remote task list files, unless you explicitly override the default by specifying the resource requirement string on the command line.

User-level resource requirements

You may have applications that you need to control yourself. Perhaps your administrator did not set them up for load sharing for all users, or you need a non-standard setup. You can use LSF commands to find out resource names available in your system, and tell LSF about the needs of your applications. LSF stores the resource requirements for you from then on.

You can specify resource requirements when tasks are added to the user's remote task list. If the task to be added is already in the list, its resource requirements are replaced.

```
% lsrtasks + myjob/swap>=100 && cpu
```

This adds `myjob` to the remote tasks list with its resource requirements.

Task Files

There are 3 task list files that can affect a job:

- lsf.task—system-wide defaults apply to all LSF users, even across multiple clusters if MultiCluster is installed
- lsf.task.*cluster_name*—cluster-wide defaults apply to all users in the cluster
- $HOME/.lsftask—user-level defaults apply to a single user

  This file lists applications to be added to or removed from the default system lists for your jobs. Resource requirements specified in this file override those in the system lists.

The clusterwide task file is used to augment the systemwide file. The user's task file is used to augment the systemwide and clusterwide task files.

LSF combines the systemwide, clusterwide, and user-specific task lists for each user's view of the task list. In cases of conflicts, such as different resource requirements specified for the same task in different lists, the clusterwide list overrides the systemwide list, and the user-specific list overrides both.

LSF_CONFDIR/lsf.task

Systemwide task list applies to all clusters and all users.

This file is used in a MultiCluster environment.

LSF_CONFDIR/lsf.task.*cluster_name*

Clusterwide task list applies to all users in the same cluster.

$HOME/.lsftask

User task list, one per user, applies only to the specific user. This file is automatically created in the user's home directory whenever a user first updates his task lists using the lsrtasks or lsltasks commands. For details about task eligibility lists, see the man page ls_task(3).

Permissions

Only the LSF administrator can modify the systemwide task list(lsf.task) and the clusterwide task list(lsf.task.*cluster_name*).

A user can modify his own task list(.lsftask) with the lsrtasks and lsltasks commands. See the man pages lsrtasks(1) and lsltasks(1) for more details.

Format of Task Files

Each file consists of two sections, `LocalTasks` and `RemoteTasks`. For example:

```
Begin LocalTasks
ps
hostname
uname
crontab
End LocalTasks
```

```
Begin RemoteTasks
+ "newjob/mem>25"
+ "verilog/select[type==any && swp>100]"
make/cpu
nroff/-
End RemoteTasks
```

Tasks are listed one per line. Each line in a section consists of a task name, and, for the `RemoteTasks` section, an optional resource requirement string separated by a slash (`/`).

A plus sign (+) or a minus sign (-) can optionally precede each entry. If no `+` or `-` is specified, `+` is assumed.

A `+` before a task name means adding a new entry (if non-existent) or replacing an entry (if already existent) in the task list. A `-` before a task name means removing an entry from the application's task lists if it was already created by reading higher level task files.

LocalTasks Section

The section starts with `Begin LocalTasks` and ends with `End LocalTasks`.

This section lists tasks that are not eligible for remote execution, either because they are trivial tasks or because they need resources on the local host.

RemoteTasks Section

The section starts with `Begin RemoteTasks` and ends with `End RemoteTasks`.

This section lists tasks that are eligible for remote execution. You can associate resource requirements with each task name.

See `lsfintro`(1) for a description of the resource requirement string. If the resource requirement string is not specified for a remote task, the default is `"select[type==local] order[r15s:pg]"`.

SEE ALSO

lsfintro(1), lsrtasks(1), lsltasks(1), ls_task(3), lsf.conf(5)

setup.config

Contents

- "About setup.config" on page 624
- "Parameters" on page 625

About setup.config

The `setup.config` file contains options for Platform LSF License Scheduler installation and configuration for systems without Platform LSF. You only need to edit this file if you are installing License Scheduler as a standalone product without LSF.

Template location

A template `setup.config` is included in the License Scheduler installation script tar file and is located in the directory created when you uncompress and extract the installation script tar file. Edit the file and uncomment the options you want in the template file. Replace the example values with your own settings to specify the options for your new License Scheduler installation.

Important The sample values in the setup.config template file are examples only. They are not default installation values.

After the License Scheduler installation, the `setup.config` containing the options you specified is located in `LS_TOP/6.2/install/`.

Format

Each entry in `setup.config` has the form:

```
NAME="STRING1 STRING2 ..."
```

The equal sign `=` must follow each `NAME` even if no value follows and there should be no spaces around the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Blank lines and lines starting with a pound sign (`#`) are ignored.

Parameters

- "LS_ADMIN"
- "LS_HOSTS"
- "LS_LICENSE_FILE"
- "LS_LMSTAT_PATH"
- "LS_TOP"

LS_ADMIN

Syntax **LS_ADMIN="***user_name* [*user_name* ...]**"**

Description Lists the License Scheduler administrators. The first user account name in the list is the primary License Scheduler administrator.

The primary License Scheduler administrator account is typically named lsadmin.

CAUTION **You should *not* configure the root account as the primary License Scheduler administrator.**

Valid Values User accounts for License Scheduler administrators must exist on all hosts using License Scheduler prior to installation.

Example `LS_ADMINS="lsadmin user1 user2"`

Default The user running the License Scheduler installation script.

LS_HOSTS

Syntax **LS_HOSTS="***host_name* [*host_name* ...]**"**

Description Defines a list of hosts that are candidates to become License Scheduler master hosts. Provide at least one host from which the License Scheduler daemon will run.

Valid Values Any valid License Scheduler host name.

Example `LS_HOSTS="host_name1 host_name2"`

Default The local host in which the License Scheduler installation script is running.

LS_LICENSE_FILE

Syntax **LS_LICENSE_FILE="/***path***/***license_file***"**

Description Defines the full path to, and name of the License Scheduler license file.

Valid Values Any valid file name and directory path.

Example `LS_LICENSE_FILE="/usr/share/ls/conf/license.dat"`

Default `$LS_TOP/conf/license.dat`

LS_LMSTAT_PATH

Syntax **LS_LMSTAT_PATH="/***path***"**

Description Defines the full path to the lmstat program. License Scheduler uses `lmstat` to gather the FLEXnet license information for scheduling. This path does not include the name of the `lmstat` program itself.

Example `LS_LMSTAT_PATH="/usr/bin"`

Default The installation script attempts to find a working copy of `lmstat` on the current system. If it is unsuccessful, the path is set as blank (`""`).

LS_TOP

Syntax **`LS_TOP="/`*`path`*`"`**

Description Defines the full path to the top level License Shceduler installation directory.

Valid Values Must be an absolute path to a shared directory that is accessible to all hosts using License Scheduler. Cannot be the root directory (`/`).

Recommended Value The file system containing LS_TOP must have enough disk space for all host types (approximately 300 MB per host type).

Example `LS_TOP="/usr/share/ls"`

Default None—required variable

SEE ALSO

`install.config`(5)

slave.config

Contents

- "About slave.config" on page 628
- "Parameters" on page 629

About slave.config

Dynamically added LSF hosts that will not be master candidates are *slave hosts*. Each dynamic slave host has its own LSF binaries and local `lsf.conf` and shell environment scripts (`cshrc.lsf` and `profile.lsf`). You must install LSF on each slave host.

The `slave.config` file contains options for installing and configuring a slave host that can be dynamically added or removed.

Use `lsfinstall -s -f slave.config` to install LSF using the options specified in `slave.config`.

Template Location

A template `slave.config` is located in the installation script directory created when you extract the LSF installation script tar file. Edit the file and uncomment the options you want in the template file. Replace the example values with your own settings to specify the options for your new LSF installation.

Important The sample values in the slave.config template file are examples only. They are not default installation values.

Format

Each entry in `slave.config` has the form:

```
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each `NAME` even if no value follows and there should be no spaces around the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Blank lines and lines starting with a pound sign (#) are ignored.

Parameters

- "LSF_ADMINS"
- "LSF_LIM_PORT"
- "LSF_SERVER_HOSTS"
- "LSF_TARDIR"
- "LSF_LOCAL_RESOURCES"
- "LSF_TOP"

LSF_ADMINS

Syntax **LSF_ADMINS="***user_name* [*user_name* ...]**"**

Description Required. Lists the LSF administrators. The first user account name in the list is the primary LSF administrator in `lsf.cluster.`*cluster_name*.

The LSF administrator accounts must exist on all hosts in the LSF cluster before installing LSF

The primary LSF administrator account is typically named lsfadmin. It owns the LSF configuration files and log files for job events. It also has permission to reconfigure LSF and to control batch jobs submitted by other users. It typically does not have authority to start LSF daemons. Unless an `lsf.sudoers` file exists to grant LSF administrators permission, only root has permission to start LSF daemons.

CAUTION **You should *not* configure the root account as the primary LSF administrator.**

Valid Values User accounts for LSF administrators must exist on all hosts in the LSF cluster before running `lsfinstall`.

Example `LSF_ADMINS="lsfadmin user1 user2"`

Default None—required variable

LSF_LIM_PORT

Syntax **LSF_LIM_PORT="***port_number***"**

Description Optional. TCP service port for slave host to use for communication with the LSF master LIM daemon. Use the same port number as LSF_LIM_PORT in `lsf.conf` on the master host.

If not specified, the default `LSF_LIM_PORT="6879"` is used.

Default Undefined

LSF_SERVER_HOSTS

Syntax **LSF_SERVER_HOSTS="***host_name* [*host_name* ...]**"**

Description Optional for shared installation. Required for non-shared installation.

Lists the hosts in the cluster to be set up as LSF server hosts.

Specify a list of host names two ways:

- Host names separated by spaces

- Name of a file containing a list of host names, one host per line.

Valid Values Any valid LSF host name

Examples

- List of host names:

  `LSF_SERVER_HOSTS="hosta hostb hostc hostd"`

- Host list file:

  `LSF_SERVER_HOSTS=:lsf_server_hosts`

  The file `lsf_server_hosts` contains a list of hosts:

  ```
  hosta
  hostb
  hostc
  hostd
  ```

Default The local host where `lsfinstall` is running

LSF_TARDIR

Syntax **`LSF_TARDIR="/`*`path`*`"`**

Description Optional. Full path to the directory containing the LSF distribution tar files.

Example `LSF_TARDIR="/usr/local/lsf_distrib"`

Default The parent directory of the current working directory where `lsfinstall` is running (`../current_directory`)

LSF_LOCAL_RESOURCES

Syntax **`LSF_LOCAL_RESOURCES="`***resource ...***`"`**

Description Optional. Defines instances of local resources residing on the slave host.

- For numeric resources, define name-value pairs:

  `"[resourcemap` *`value`***`*`***`resource_name`***`]"`**

- For Boolean resources, define the resource name in the form:

  `"[resource` *`resource_name`***`]"`**

When the slave host calls the master host to add itself, it also reports its local resources. The local resources to be added must be defined in `lsf.shared`.

If the same resource is already defined in `lsf.shared` as `default` or `all`, it cannot be added as a local resource. The shared resource overrides the local one.

LSF_LOCAL_RESOURCES is usually set in the `slave.config` file during installation. If LSF_LOCAL_RESOURCES are already defined in a local `lsf.conf` on the slave host, `lsfinstall` does not add resources you define in LSF_LOCAL_RESOURCES in `slave.config`. You should not have duplicate LSF_LOCAL_RESOURCES entries in lsf.conf. If local resources are defined more than once, only the last definition is valid.

IMPORTANT **Resources must already be mapped to hosts in the ResourceMap section of lsf.cluster.*cluster_name*. If the ResourceMap section does not exist, local resources are not added.**

Example `LSF_LOCAL_RESOURCES="[resourcemap 1*verilog] [resource linux]"`

Default None—optional variable

LSF_TOP

Syntax **LSF_TOP="/***path***"**

Description Required. Full path to the top-level LSF installation directory.

Valid value Must be an absolute path to a local directory on the slave host.

Cannot be the root directory (/).

Recommended value The file system containing LSF_TOP must have enough disk space for all host types (approximately 300 MB per host type).

Example `LSF_TOP="/usr/local/lsf"`

Default None—required variable

SEE ALSO

`lsfinstall`(8), `install.config`(5), `lsf.cluster`(5), `lsf.sudoers`(5)

win_install.config

Contents

- "About win_install.config" on page 634
- "Parameters" on page 635

About win_install.config

The `win_install.config` file contains options for Platform LSF for Windows installation and configuration using the silent install option. Use `setup /I:win_install.config` to install LSF using the options specified in `win_install.config`.

Template location

A template `win_install.config` is included in the LSF for Windows installation file `lsf6.2_win.exe`.

To edit win_install.config

1. Extract `win_install.config` from the `lsf6.2_win.exe` installation file.
2. Edit the file and uncomment the options you want in the template file.

   Replace the example values with your own settings to specify the options for your new LSF installation.

Important The sample values in the win_install.config template file are *examples* only. They are not default installation values.

Format

Each entry in `win_install.config` has the form:

```
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each `NAME` even if no value follows and there should be no spaces around the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Blank lines and lines starting with a pound sign (#) are ignored.

Parameters

Required parameters

The following parameters are required and must be defined:

- "INSTALL_OPTION"
- "LSF_TOP"
- "LSF_CLUSTER_NAME"
- "LOCAL_DIR"
- "SERVICE_ACCT"

Optional parameters

The following parameters are optional:

- "LIM_PORT"
- "LSF_CLIENTS"
- "LSF_DYNAMIC_SERVERS"
- "LSF_SERVERS"
- "SERVER_HOST"

If LSF_SERVERS, LSF_DYNAMIC_SERVERS, and LSF_CLIENTS are not defined, the local host is installed as LSF_SERVER.

INSTALL_OPTION

Syntax **INSTALL_OPTION="***installation_type***"**

Description Required—defines the intended action of this particular setup session.

Valid Values

- `ADD_HOST`
- `UPGRADE_HOST`
- `UNINSTALL_HOST`
- `UNINSTALL_ORPHAN_HOST`

Use UNINSTALL_ORPHAN_HOST if hosts are left over after using UNINSTALL_HOST. This can happen when you uninstall a cluster before removeing LSF from all dynamic host.s When using UNINSTALL_ORPHAN_HOST, only LSF_CLUSTER_NAME is required. LOCAL_DIR, LSF_SERVERS, LSF_CLIENTS, and LSF_DYNAMIC_SERVERS are optional. Other parameters are ignored.

Example `INSTALL_OPTION="UPGRADE_HOST"`

Default None—required parameter

LIM_PORT

Syntax **LIM_PORT=***integer*

Description Optional—defines the port number for adding non-shared hosts to the cluster. You must also set SERVER_HOST. Without SERVER_HOST set, LIM_PORT is ignored.

Valid Values Must be an unused port number.

Example `LIM_PORT=6879`

Default None

See also SERVER_HOST

LOCAL_DIR

Syntax **LOCAL_DIR=***path*

Description Required—sets the local directory for the root of the path to the machine-dependent LSF files. Must be an absolute path to a local (non-shared) directory. Cannot be the root directory (`\\server_name`).

Example `LOCAL_DIR="C:\lsf_6.2_cluster"`

Default None—required parameter

LSF_CLIENTS

Syntax **LSF_CLIENTS="***host_name* | **:***host_list_file* [*host_name* | **:***host_list_file* ...]**"**

Description Optional—lists the hosts in the cluster to be set up as LSF client hosts.

After installation, you must manually edit `lsf.cluster.`*cluster_name* to include the correct host model and type of each static client listed in LSF_CLIENTS. This will enable automatic host type and model detection when the client host LIM starts.

Valid Values Any valid LSF host name, or any file containing a list of valid LSF host name. The file containing the list cannot have any white spaces and must list one host per line.

Examples

- `LSF_CLIENTS="hostk hostk"`
- `LSF_CLIENTS=":lsf_client_hosts1 :lsf_client_hosts2"`

  where `lsf_client_hosts1` is a text file containing the following:

  ```
  hostk
  hostl
  ```

Default None

LSF_DYNAMIC_SERVERS

Syntax **LSF_DYNAMIC_SERVERS="***host_name* | **:***host_list_file* [*host_name* | **:***host_list_file* ...]**"**

Description Optional—lists the hosts in the cluster to be set up as dynamic LSF server hosts.

Valid Values Any valid LSF host name, or any file containing a list of valid LSF host name. The file containing the list cannot have any white spaces and must list one host per line.

Examples

- `LSF_DYNAMIC_SERVERS="hostf hostg hosth hosti hostj"`
- `LSF_DYNAMIC_SERVERS=":lsf_dyn_server_hosts1 :lsf_dyn_server_hosts2"`

  where `lsf_dyn_server_hosts1` is a text file containing the following:

  ```
  hostf
  hostg
  ```

  and `lsf_dyn_server_hosts2` is a text file containing the following:

  ```
  hosth
  hosti
  hostj
  ```

Default None

LSF_SERVERS

Syntax **LSF_SERVERS="***host_name* | **:***host_list_file* [*host_name* | **:***host_list_file* ...]**"**

Description Optional—lists the hosts in the cluster to be set up as LSF server hosts. The first host in the list becomes the LSF master host in `lsf.cluster.`*cluster_name*.

Valid Values Any valid LSF host name, or any file containing a list of valid LSF host name. The file containing the list cannot have any white spaces and must list one host per line.

Examples

- `LSF_SERVERS="hosta hostb hostc hostd hoste"`

  `hosta` is the LSF master host.
- `LSF_SERVERS=":lsf_server_hosts1 :lsf_server_hosts2"`

  where `lsf_server_hosts1` is a text file containing the following:

  ```
  hosta
  hostb
  ```

  and `lsf_server_hosts2` is a text file containing the following:

  ```
  hostc
  hostd
  hoste
  ```

  `hosta` is the LSF master host.

Default The local host where `setup` is running

LSF_CLUSTER_NAME

Syntax **LSF_CLUSTER_NAME="***cluster_name***"**

Description Required—defines the name of the LSF cluster. Do not use an LSF host name.

Valid Values Any alphanumeric string containing no more than 39 characters. The name cannot contain white spaces.

Recommended Value You should not use a valid host name as the cluster name, but the same general principles apply to naming your cluster as naming hosts.

Example `LSF_CLUSTER_NAME="cluster1"`

Default None—required parameter

LSF_TOP

Syntax **LSF_TOP="***server_name******path***"**

Description Required—defines the top-level LSF installation directory.

Valid Values Must be an absolute path to a shared directory that is accessible to all Windows hosts in the cluster. Cannot be the root directory (`\\server_name`).

Recommended Value The file system containing LSF_TOP must have enough disk space for all host types (approximately 300 MB per host type).

Example `LSF_TOP="\\hostA\LSF_6.2"`

Default None—required parameter

SERVER_HOST

Syntax **SERVER_HOST="***server_domain***"**

Description Optional—defines the non-shared hosts to add to the cluster. You must also set LIM_PORT. Without LIM_PORT set, SERVER_HOST is ignored.

Valid Values Must be a valid domain server name.

Example `SERVER_HOST="hosta.example.com"`

Default None

See also LIM_PORT

SERVICE_ACCT

Syntax **SERVICE_ACCT="**[*domain*****]*account_name***"**

Description Required—defines the user account that the LSF daemons run from.

Valid Values Must be a valid Windows user account.

Example `SERVICE_ACCT="DOMAINA\user1"`

Default None—required parameter

SEE ALSO

`lsf.cluster`(5)

PART

IV

Troubleshooting

Troubleshooting and Error Messages

Contents

- “Shared File Access” on page 642
- “Common LSF Problems” on page 643
- “Error Messages” on page 646

Shared File Access

A frequent problem is non-accessible files due to a non-uniform file space. If a task is run on a remote host where a file it requires cannot be accessed using the same name, an error results. Almost all interactive LSF commands fail if the user's current working directory cannot be found on the remote host.

Shared files on UNIX

If you are running NFS, rearranging the NFS mount table may solve the problem. If your system is running the `automount` server, LSF tries to map the filenames, and in most cases it succeeds. If shared mounts are used, the mapping may break for those files. In such cases, specific measures need to be taken to get around it.

The automount maps must be managed through NIS. When LSF tries to map filenames, it assumes that automounted file systems are mounted under the `/tmp_mnt` directory.

Shared files on Microsoft Windows

To share files among Windows machines, set up a share on the server and access it from the client. You can access files on the share either by specifying a UNC path (`\\server\share\path`) or connecting the share to a local drive name and using a `drive:\path` syntax. Using UNC is recommended because drive mappings may be different across machines, while UNC allows you to unambiguously refer to a file on the network.

Shared files across UNIX and Windows

For file sharing across UNIX and Windows, you require a third party NFS product on Windows to export directories from Windows to UNIX.

Common LSF Problems

This section lists some common problems with LSF jobs. Most problems are due to incorrect installation or configuration. Check the `mbatchd` and `sbatchd` error log files; often the log message points directly to the problem.

The section also includes some common problems with the LIM, the RES and interactive applications.

LIM dies quietly

Run the following command to check for errors in the LIM configuration files.

`% lsadmin ckconfig -v`

This displays most configuration errors. If this does not report any errors, check in the LIM error log.

LIM unavailable

Sometimes the LIM is up, but executing the `lsload` command prints the following error message:

```
Communication time out.
```

If the LIM has just been started, this is normal, because the LIM needs time to get initialized by reading configuration files and contacting other LIMs.

If the LIM does not become available within one or two minutes, check the LIM error log for the host you are working on.

When the local LIM is running but there is no master LIM in the cluster, LSF applications display the following message:

```
Cannot locate master LIM now, try later.
```

Check the LIM error logs on the first few hosts listed in the `Host` section of the `lsf.cluster.`*cluster_name* file. If LSF_MASTER_LIST is defined in `lsf.conf`, check the LIM error logs on the hosts listed in this parameter instead.

Master LIM is down

Sometimes the master LIM is up, but executing the `lsload` or `lshosts` command prints the following error message:

```
Master LIM is down; try later
```

If the `/etc/hosts` file on the host where the master LIM is running is configured with the host name assigned to the loopback IP address (127.0.0.1), LSF client LIMs cannot contact the master LIM. When the master LIM starts up, it sets its official host name and IP address to the loopback address. Any client requests will get the master LIM address as 127.0.0.1, and try to connect to it, and in fact will try to access itself.

Check the IP configuration of your master LIM in `/etc/hosts`. The following example incorrectly sets the master LIM IP address to the loopback address:

```
127.0.0.1         localhost      myhostname
```

The following example correctly sets the master LIM IP address:

```
127.0.0.1         localhost
192.168.123.123   myhostname
```

RES does not start

Check the RES error log.

UNIX If the RES is unable to read the `lsf.conf` file and does not know where to write error messages, it logs errors into `syslog(3)`.

Windows If the RES is unable to read the `lsf.conf` file and does not know where to write error messages, it logs errors into `C:\temp`.

User permission denied

If remote execution fails with the following error message, the remote host could not securely determine the user ID of the user requesting remote execution.

```
User permission denied.
```

Check the RES error log on the remote host; this usually contains a more detailed error message.

If you are not using an identification daemon (LSF_AUTH is not defined in the `lsf.conf` file), then all applications that do remote executions must be owned by root with the `setuid` bit set. This can be done as follows.

```
% chmod 4755 filename
```

If the binaries are on an NFS-mounted file system, make sure that the file system is not mounted with the `nosuid` flag.

If you are using an identification daemon (defined in the `lsf.conf` file by LSF_AUTH), `inetd` must be configured to run the daemon. The identification daemon must not be run directly.

If LSF_USE_HOSTEQUIV=Y is defined in the `lsf.conf` file, check if `/etc/hosts.equiv` or `HOME/.rhosts` on the destination host has the client host name in it. Inconsistent host names in a name server with `/etc/hosts` and `/etc/hosts.equiv` can also cause this problem.

On SGI hosts running a name server, you can try the following command to tell the host name lookup code to search the `/etc/hosts` file before calling the name server.

```
% setenv HOSTRESORDER "local,nis,bind"
```

Non-uniform file name space

A command may fail with the following error message due to a non-uniform file name space.

```
chdir(...) failed: no such file or directory
```

You are trying to execute a command remotely, where either your current working directory does not exist on the remote host, or your current working directory is mapped to a different name on the remote host.

If your current working directory does not exist on a remote host, you should not execute commands remotely on that host.

On UNIX If the directory exists, but is mapped to a different name on the remote host, you have to create symbolic links to make them consistent.

LSF can resolve most, but not all, problems using `automount`. The automount maps must be managed through NIS. Follow the instructions in your Release Notes for obtaining technical support if you are running automount and LSF is not able to locate directories on remote hosts.

Batch daemons die quietly

First, check the `sbatchd` and `mbatchd` error logs. Try running the following command to check the configuration.

```
% badmin ckconfig
```

This reports most errors. You should also check if there is any email from LSF in the LSF administrator's mailbox. If the `mbatchd` is running but the `sbatchd` dies on some hosts, it may be because `mbatchd` has not been configured to use those hosts.

See "Host not used by LSF" on page 645.

sbatchd starts but mbatchd does not

Check whether LIM is running. You can test this by running the `lsid` command. If LIM is not running properly, follow the suggestions in this chapter to fix the LIM first. You should make sure that all hosts use the same `lsf.conf` file. Note that it is possible that `mbatchd` is temporarily unavailable because the master LIM is temporarily unknown, causing the following error message.

```
sbatchd: unknown service
```

Check whether services are registered properly. See *Administering Platform LSF* for information about registering LSF services.

Host not used by LSF

If you configure a list of server hosts in the `Host` section of the `lsb.hosts` file, `mbatchd` allows `sbatchd` to run only on the hosts listed. If you try to configure an unknown host as a `HOSTS` definition for a queue in the `lsb.queues` file, `mbatchd` logs the following message.

```
mbatchd on host: LSB_CONFDIR/cluster/configdir/file(line #):
Host hostname is not used by lsbatch;

ignored
```

If you try to configure an unknown host in the `HostGroup` or `HostPartition` sections of the `lsb.hosts` file, you also see the message.

If you start `sbatchd` on a host that is not known by `mbatchd`, `mbatchd` rejects the `sbatchd`. The `sbatchd` logs the following message and exits.

```
This host is not used by lsbatch system.
```

Both of these errors are most often caused by not running the following commands, in order, after adding a host to the configuration.

```
lsadmin reconfig
badmin reconfig
```

You must run both of these before starting the daemons on the new host.

Error Messages

The following error messages are logged by the LSF daemons, or displayed by the following commands.

```
lsadmin ckconfig
badmin ckconfig
```

General errors

The messages listed in this section may be generated by any LSF daemon.

`can't open file: error`

The daemon could not open the named file for the reason given by *error*. This error is usually caused by incorrect file permissions or missing files. All directories in the path to the configuration files must have execute (`x`) permission for the LSF administrator, and the actual files must have read (`r`) permission. Missing files could be caused by incorrect path names in the `lsf.conf` file, running LSF daemons on a host where the configuration files have not been installed, or having a symbolic link pointing to a nonexistent file or directory.

`file(line): malloc failed`

Memory allocation failed. Either the host does not have enough available memory or swap space, or there is an internal error in the daemon. Check the program load and available swap space on the host; if the swap space is full, you must add more swap space or run fewer (or smaller) programs on that host.

`auth_user: getservbyname(ident/tcp) failed: error; ident must be registered in services`

LSF_AUTH=ident is defined in the `lsf.conf` file, but the `ident/tcp` service is not defined in the services database. Add `ident/tcp` to the services database, or remove LSF_AUTH from the `lsf.conf` file and `setuid root` those LSF binaries that require authentication.

`auth_user: operation(<host>/<port>) failed: error`

LSF_AUTH=ident is defined in the `lsf.conf` file, but the LSF daemon failed to contact the `identd` daemon on host. Check that `identd` is defined in `inetd.conf` and the `identd` daemon is running on host.

`auth_user: Authentication data format error (rbuf=<data>) from <host>/<port>`

`auth_user: Authentication port mismatch (...) from <host>/<port>`

LSF_AUTH=ident is defined in the `lsf.conf` file, but there is a protocol error between LSF and the ident daemon on *host*. Make sure the ident daemon on the host is configured correctly.

`userok: Request from bad port (<port_number>), denied`

LSF_AUTH is not defined, and the LSF daemon received a request that originates from a non-privileged port. The request is not serviced.

Set the LSF binaries (for example, `lsrun`) to be owned by root with the `setuid` bit set, or define LSF_AUTH=ident and set up an ident server on all hosts in the cluster. If the binaries are on an NFS-mounted file system, make sure that the file system is not mounted with the `nosuid` flag.

userok: Forged username suspected from <*host*>/<port>: <*claimed_user*>/<*actual_user*>

The service request claimed to come from user *claimed_user* but ident authentication returned that the user was actually *actual_user*. The request was not serviced.

userok: ruserok(<host>,<uid>) failed

LSF_USE_HOSTEQUIV=Y is defined in the lsf.conf file, but *host* has not been set up as an equivalent host (see /etc/host.equiv), and user *uid* has not set up a .rhosts file.

init_AcceptSock: RES service(res) not registered, exiting

init_AcceptSock: res/tcp: unknown service, exiting

initSock: LIM service not registered.

initSock: Service lim/udp is unknown. Read LSF Guide for help

get_ports: <serv> service not registered

The LSF services are not registered. See *Administering Platform LSF* for information about configuring LSF services.

init_AcceptSock: Can't bind daemon socket to port <port>: error, exiting

init_ServSock: Could not bind socket to port <port>: error

These error messages can occur if you try to start a second LSF daemon (for example, RES is already running, and you execute RES again). If this is the case, and you want to start the new daemon, kill the running daemon or use the lsadmin or badmin commands to shut down or restart the daemon.

Configuration errors

The messages listed in this section are caused by problems in the LSF configuration files. General errors are listed first, and then errors from specific files.

file(line): Section name expected after Begin; ignoring section

file(line): Invalid section name name; ignoring section

The keyword begin at the specified line is not followed by a section name, or is followed by an unrecognized section name.

file(line): section section: Premature EOF

The end of file was reached before reading the end section line for the named section.

file(line): keyword line format error for section section; Ignore this section

The first line of the section should contain a list of keywords. This error is printed when the keyword line is incorrect or contains an unrecognized keyword.

file(line): values do not match keys for section section; Ignoring line

The number of fields on a line in a configuration section does not match the number of keywords. This may be caused by not putting () in a column to represent the default value.

file: HostModel section missing or invalid

file: Resource section missing or invalid

file: HostType section missing or invalid

The `HostModel`, `Resource`, or `HostType` section in the `lsf.shared` file is either missing or contains an unrecoverable error.

`file(line): Name name reserved or previously defined. Ignoring index`

The name assigned to an external load index must not be the same as any built-in or previously defined resource or load index.

`file(line): Duplicate clustername name in section cluster. Ignoring current line`

A cluster name is defined twice in the same `lsf.shared` file. The second definition is ignored.

`file(line): Bad cpuFactor for host model model. Ignoring line`

The CPU factor declared for the named host model in the `lsf.shared` file is not a valid number.

`file(line): Too many host models, ignoring model name`

You can declare a maximum of 127 host models in the `lsf.shared` file.

`file(line): Resource name name too long in section resource. Should be less than 40 characters. Ignoring line`

The maximum length of a resource name is 39 characters. Choose a shorter name for the resource.

`file(line): Resource name name reserved or previously defined. Ignoring line.`

You have attempted to define a resource name that is reserved by LSF or already defined in the `lsf.shared` file. Choose another name for the resource.

`file(line): illegal character in resource name: name, section resource. Line ignored.`

Resource names must begin with a letter in the set [a-zA-Z], followed by letters, digits or underscores [a-zA-Z0-9_].

LIM messages

The following messages are logged by the LIM:

`main: LIM cannot run without licenses, exiting`

The LSF software license key is not found or has expired. Check that FLEXnet is set up correctly, or contact Platform support at support@platform.com.

`main: Received request from unlicensed host <host>/<port>`

LIM refuses to service requests from hosts that do not have licenses. Either your LSF license has expired, or you have configured LSF on more hosts than your license key allows.

`initLicense: Trying to get license for LIM from source <LSF_CONFDIR/license.dat>`

`getLicense: Can't get software license for LIM from license file <LSF_CONFDIR/license.dat>: feature not yet available.`

Your LSF license is not yet valid. Check whether the system clock is correct.

`findHostbyAddr/<proc>: Host <host>/<port> is unknown by <myhostname>`

`function: Gethostbyaddr_(<host>/<port>) failed: error`

`main: Request from unknown host <host>/<port>: error` `function: Received request from non-LSF host <host>/<port>`

The daemon does not recognize *host* as a Platform LSF host. The request is not serviced. These messages can occur if *host* was added to the configuration files, but not all the daemons have been reconfigured to read the new information. If the problem still occurs after reconfiguring all the daemons, check whether the host is a multi-addressed host. See *Administering Platform LSF* for information about working with multi-addressed hosts.

`rcvLoadVector: Sender (<host>/<port>) may have different config?`

`MasterRegister: Sender (host) may have different config?`

LIM detected inconsistent configuration information with the sending LIM. Run the following command so that all the LIMs have the same configuration information.

```
% lsadmin reconfig
```

Note any hosts that failed to be contacted.

`rcvLoadVector: Got load from client-only host <host>/<port>. Kill LIM on <host>/<port>`

A LIM is running on a Platform LSF client host. Run the following command, or go to the client host and kill the LIM daemon.

```
% lsadmin limshutdown host
```

`saveIndx: Unknown index name <name> from ELIM`

LIM received an external load index name that is not defined in the `lsf.shared` file. If name is defined in `lsf.shared`, reconfigure the LIM. Otherwise, add name to the `lsf.shared` file and reconfigure all the LIMs.

`saveIndx: ELIM over-riding value of index <name>`

This is a warning message. The ELIM sent a value for one of the built-in index names. LIM uses the value from ELIM in place of the value obtained from the kernel.

`getusr: Protocol error numIndx not read (cc=num): error`

`getusr: Protocol error on index number (cc=num): error`

Protocol error between ELIM and LIM. See *Administering Platform LSF* for a description of the ELIM and LIM protocols.

RES messages

These messages are logged by the RES.

`doacceptconn: getpwnam(<username>@<host>/<port>) failed: error`

`doacceptconn: User <username> has uid <uid1> on client host <host>/<port>, uid <uid2> on RES host; assume bad user`

`authRequest: username/uid <userName>/<uid>@<host>/<port> does not exist`

`authRequest: Submitter's name <clname>@<clhost> is different from name <lname> on this host`

RES assumes that a user has the same userID and username on all the LSF hosts. These messages occur if this assumption is violated. If the user is allowed to use LSF for interactive remote execution, make sure the user's account has the same user ID and user name on all LSF hosts.

`doacceptconn: root remote execution permission denied` `authRequest: root job submission rejected`

Root tried to execute or submit a job but LSF_ROOT_REX is not defined in the `lsf.conf` file.

`resControl: operation permission denied, uid = <uid>`

The user with user ID *uid* is not allowed to make RES control requests. Only the LSF administrator, or root if LSF_ROOT_REX is defined in `lsf.conf`, can make RES control requests.

`resControl: access(respath, X_OK): error`

The RES received a reboot request, but failed to find the file `respath` to re-execute itself. Make sure `respath` contains the RES binary, and it has execution permission.

LSF messages

The following messages are logged by the `mbatchd` and `sbatchd` daemons:

`renewJob: Job <jobId>: rename(<from>,<to>) failed: error`

`mbatchd` failed in trying to re-submit a rerunnable job. Check that the file *from* exists and that the LSF administrator can rename the file. If *from* is in an AFS directory, check that the LSF administrator's token processing is properly setup

See *Administering Platform LSF* for information about installing on AFS.

```
logJobInfo_: fopen(<logdir/info/jobfile>) failed: error
logJobInfo_: write <logdir/info/jobfile> <data> failed: error
logJobInfo_: seek <logdir/info/jobfile> failed: error
logJobInfo_: write <logdir/info/jobfile> xdrpos <pos> failed: error
logJobInfo_: write <logdir/info/jobfile> xdr buf len <len> failed: error
logJobInfo_: close(<logdir/info/jobfile>) failed: error
rmLogJobInfo: Job <jobId>: can't unlink(<logdir/info/jobfile>): error
rmLogJobInfo_: Job <jobId>: can't stat(<logdir/info/jobfile>): error
readLogJobInfo: Job <jobId> can't open(<logdir/info/jobfile>): error
start_job: Job <jobId>: readLogJobInfo failed: error
readLogJobInfo: Job <jobId>: can't read(<logdir/info/jobfile>) size size: error
initLog: mkdir(<logdir/info>) failed: error
<fname>: fopen(<logdir/file> failed: error
getElogLock: Can't open existing lock file <logdir/file>: error
getElogLock: Error in opening lock file <logdir/file>: error
releaseElogLock: unlink(<logdir/lockfile>) failed: error
touchElogLock: Failed to open lock file <logdir/file>: error
touchElogLock: close <logdir/file> failed: error
```

`mbatchd` failed to create, remove, read, or write the log directory or a file in the log directory, for the reason given in *error*. Check that LSF administrator has read, write, and execute permissions on the `logdir` directory.

If `logdir` is on AFS, check that the instructions in *Administering Platform LSF* have been followed. Use the `fs ls` command to verify that the LSF administrator owns `logdir` and that the directory has the correct ACL.

`replay_newjob: File <logfile> at line <line>: Queue <queue> not found, saving to queue <lost_and_found>`

`replay_switchjob: File <logfile> at line <line>: Destination queue <queue> not found, switching to queue <lost_and_found>`

When `mbatchd` was reconfigured, jobs were found in *queue* but that queue is no longer in the configuration.

`replay_startjob: JobId <jobId>: exec host <host> not found, saving to host <lost_and_found>`

When `mbatchd` was reconfigured, the event log contained jobs dispatched to host, but that host is no longer configured to be used by LSF.

`do_restartReq: Failed to get hData of host <host_name>/<host_addr>`

`mbatchd` received a request from `sbatchd` on host *host_name*, but that host is not known to `mbatchd`. Either the configuration file has been changed but `mbatchd` has not been reconfigured to pick up the new configuration, or *host_name* is a client host but the `sbatchd` daemon is running on that host. Run the following command to reconfigure the `mbatchd` or kill the `sbatchd` daemon on *host_name*.

```
% badmin reconfig
```

Index

Symbols

.lsftask file 617
.rhosts file 644
/etc/hosts file 644
/etc/hosts.equiv file 644

A

ABS_RUNLIMIT, lsb.params file 375
absolute path, lsfinstall options 202
account mapping in MultiCluster 470
ACCT_ARCHIVE_AGE, lsb.params file 375
ACCT_ARCHIVE_SIZE, lsb.params file 376
ACCT_ARCHIVE_TIME, lsb.params file 376
Active status, bqueues 111
ACTIVE WINDOW, bsla 143
Active:Missed status, bsla 144
Active:Ontime status, bsla 144
ADJUST_DURATION, lsf.cluster file 481
ADMIN
  blinfo output 84
  lsclusters 199
  lsf.licensescheduler file Parameters section, description 578
ADMIN ACTION COMMENT
  bhosts -l 54
  bqueues -l 121
administrator, lsfinstall command 203
ADMINISTRATORS
  bqueues -l 119
  lsb.queues file 400
  lsf.cluster file 489
ALLOCATION, lsf.licensescheduler file Feature section, description 587
ARCHITECTURE, lsf.shared file 602
ARRAY_SPEC, bjobs -A 68
automatic time-based configuration
  lsb.hosts 365
  lsb.params 396
  lsb.queues 430
  lsb.resources 455
  lsb.users 472
automount, NFS (Network File System) 642

B

bacct 13
BACKFILL
  bqueues -l 117
  lsb.queues file 400
badmin 24
bbot 35
bchkpnt 37
bclusters 39
bgadd 41
bgdel 42
bgroup 58
bhist 43
bhosts 49
bhpart 56
bjobs 60
bkill 70
bladmin 75
blcollect 76
bld, License Scheduler daemon 75, 78, 584
bld.license.acct file 295
blhosts 78
blimits 79
blinfo 83
blkill 87
Blocks in, lsacct 188
Blocks out, lsacct 188
blstat 88
bltasks 92
blusers 95
bmgroup 98
bmig 99
bmod 101
bparams 106
bpeek 107
bpost 108
bqc, obsolete command. *See* badmin qclose
bqueues 110
bread 123
breboot, obsolete command. *See* badmin reconfig
breconfig, obsolete command. *See* badmin reconfig
brequeue 125
bresources 127
brestart 128
bresume 130
brlainfo 132
brsrvs 139
brsvadd 134
brsvdel 138
brun 141
bsla 143
bstatus 146
bstop 148
bsub 150
BSUB_BLOCK variable 269

BSUB_QUIET variable 269
BSUB_QUIET2 variable 270
BSUB_STDERR variable 270
bswitch 176
btop 178
bugroup 180
bulk jobs, killing 71
busers 181

C

CACHE_INTERVAL, lsf.cluster file 497
ch 183
CHECKPOINT, bqueues -l 121
CHKPNT
 lsb.hosts file 354
 lsb.queues file 400
CHKPNTDIR, bqueues -l 121
CHKPNTPERIOD, bqueues -l 121
chunk jobs
 bmig 99
 bsub restrictions 151
 bswitch 176
 CHKPNT parameter in lsb.queues 400
 MIG parameter in lsb.queues 416
 rerunnable 421
CHUNK_JOB_DURATION, lsb.params file 376
CHUNK_JOB_SIZE
 bqueues -l 120
 lsb.queues file 400
CLEAN_PERIOD, lsb.params file 377
cleanup 19
CLEARCASE_DRIVE variable 270
CLEARCASE_MOUNTDIR variable 271
CLEARCASE_ROOT variable 271
Closed status, bqueues 111
CLUSTER
 bclusters 39
 blusers output 96
cluster name, lsfinstall command 203
CLUSTER_NAME, lsclusters 198
CLUSTERNAME, lsf.cluster file 497
ClusterName, lsf.shared file 600
CLUSTERS, lsf.licensescheduler file Clusters section 582
Clusters section, lsf.licensescheduler file, description 582
Command
 bhist -l 47
 bjobs -l 65
Command line, lsacct -l 188
COMMITTED_RUN_TIME_FACTOR, lsb.params file 378
COMPL_TIME, bacct -l 18
Completion time, lsacct -l 188
CONDENSE, lsb.hosts file 358
CONDENSE_PENDING_REASONS, lsb.params file 377
configurable job ID limit 385
CONTROL_ACTION, lsb.serviceclasses file 458
CORELIMIT
 bqueues -l 115
 lsb.queues file 401
CPU time
 bjobs -l 66
 lsacct 187
CPU_RADIUS, brlainfo 133
CPU_T, bacct -b 17
CPU_TIME, bhpart 57
CPU_TIME_FACTOR, lsb.params file 377
CPUF, bhosts -l 53
cpuf, lshosts 217
CPUFACTOR, lsf.shared file 602
CPULIMIT
 bqueues -l 114
 lsb.queues file 401
CPUSET_OS, brlainfo 132
Cray checkpointing 354
CREATOR, bacct -U 18
CSA (IRIX Comprehensive System Accounting), configuring and using 543
cshrc.lsf file
 description 298
 setting the LSF environment 299
CUMULATIVE_RUSAGE, LSF HPC extensions parameter 545
CURRENT LOAD, bhosts -l 54
CWD
 bacct -l 18
 bjobs -l 65
 lsacct -l 188

D

daemons, security 569
DATALIMIT
 bqueues -l 115
 lsb.queues file 402
dedicated resource. *See* exclusive resource 492
DEFAULT HOST SPECIFICATION, bqueues -l 118
Default queue indication, bqueues -l 113
DEFAULT_EXTSCHED
 bsub 157
 lsb.queues file 403
DEFAULT_HOST_SPEC
 lsb.params file 378
 lsb.queues file 403
DEFAULT_PROJECT, lsb.params file 378
DEFAULT_QUEUE, lsb.params file 378
DEMAND, blstat output 90
deprecated commands
 bqc. *See* badmin qclose
 breboot. *See* badmin reconfig
 breconfig. *See* badmin reconfig
 lslockhost. *See* lsadmin limlock
 lsreconfig. *See* lsadmin reconfig
 lsunlockhost. *See* lsadmin limunlock
DESCRIPTION
 lsb.queues file 403
 lsb.serviceclasses file 458
 lsf.shared file 604
Description, bqueues -l 113
DETECT_IDLE_JOB_AFTER, lsb.params file 379
DIRECTION, lsb.users file 470

DISABLE_UACCT_MAP, lsb.params file 379
disk space for installation 203
DISP_RES_USAGE_LIMITS, LSF HPC extensions parameter 545
DISPAT_TIME, bacct -l 17
DISPATCH_ORDER, lsb.queues file 403
DISPATCH_WINDOW
 lsb.hosts file 355
 lsb.queues file 404
DISPATCH_WINDOWS
 bhosts -l 53
 bqueues -l 118
DISPLAYS, blusers output 96
DISTRIBUTION
 blinfo output 84
 lsb.resources file HostExport section 447
 lsb.resources file SharedResourceExport section 450
 lsf.licensescheduler file Feature section, description 586
DISTRIBUTION_POLICY_VIOLATION_ACTION, lsf.licensescheduler file Parameters section, description 578
DONE
 bjobs -A 68
 bjobs -l 65
dual-core CPUs
 enabling detection 543
 license needed in lshosts -l 216
 license overuse accounting 501
 lsf.cluster_name.license.acct file 501
 ncpus in lshosts 217
DYNAMIC, lsf.licensescheduler file Feature section, description 592
dynamic slave host
 lsfinstall -s option 206
 slave.config file variables 202

E

EADMIN_TRIGGER_DURATION, lsb.params file 379
ELIM_POLL_INTERVAL, lsf.cluster file 481
ELIMARGS, lsf.cluster file 481
email, configuring on UNIX 523
ENABLE_DYNAMIC_RUSAGE, lsf.licensescheduler file Feature section, description 592
ENABLE_HIST_RUN_TIME, lsb.params file 379
ENABLE_HPC_INST, install.config file 312
ENABLE_INTERACTIVE, lsf.licensescheduler file Parameters section, description 579
ENABLE_USER_RESUME, lsb.params file 379
environment variables
 BSUB_BLOCK 269
 BSUB_QUIET 269
 BSUB_QUIET2 270
 BSUB_STDERR 270
 CLEARCASE_DRIVE 270
 CLEARCASE_MOUNTDIR 271
 CLEARCASE_ROOT 271
 LM_LICENSE_FILE 271, 580
 LS_EXEC_T 272
LS_JOBPID 272
LS_LICENSE_SERVER_feature 272
LS_SUBCWD 272
LSB_CHKPNT_DIR 272
LSB_DEBUG 273
LSB_DEBUG_CMD 273
LSB_DEBUG_MBD 273
LSB_DEBUG_NQS 273
LSB_DEBUG_SBD 273
LSB_DEBUG_SCH 273
LSB_DEFAULTPROJECT 273
LSB_DEFAULTQUEUE 274
LSB_ERESTART_USRCMD 274
LSB_EXECHOSTS 275
LSB_EXIT_PRE_ABORT 275
LSB_EXIT_REQUEUE 275
LSB_FRAMES 276
LSB_HOSTS 276
LSB_INTERACTIVE 276
LSB_JOB_STARTER 276
LSB_JOBEXIT_INFO 277
LSB_JOBEXIT_STAT 278
LSB_JOBFILENAME 278
LSB_JOBID 278
LSB_JOBINDEX 279
LSB_JOBINDEX_STEP 279
LSB_JOBNAME 279
LSB_JOBPEND 280
LSB_JOBPGIDS 280
LSB_JOBPIDS 280
LSB_MAILSIZE 280
LSB_MCPU_HOSTS 281
LSB_NQS_PORT 282
LSB_OLD_JOBID 282
LSB_OUTPUT_TARGETFAILED 282
LSB_QUEUE 282
LSB_REMOTEINDEX 283
LSB_REMOTEJID 283
LSB_RESTART 283
LSB_RESTART_PGID 283
LSB_RESTART_PID 283
LSB_SUB_CLUSTER 284
LSB_SUB_COMMAND_LINE 284
LSB_SUB_EXTSCHED_PARAM 284
LSB_SUB_JOB_ACTION_WARNING_TIME 284
LSB_SUB_JOB_WARNING_ACTION 284
LSB_SUB_PARM_FILE 284
LSB_SUSP_REASONS 284
LSB_SUSP_SUBREASONS 285
LSF_CMD_LOGDIR 285
LSF_DEBUG_CMD 286
LSF_DEBUG_LIM 286
LSF_DEBUG_RES 286
LSF_EAUTH_AUX_DATA 286
LSF_EAUTH_AUX_PASS 286
LSF_EAUTH_CLIENT 286
LSF_EAUTH_SERVER 287
LSF_EAUTH_UID 287
LSF_INTERACTIVE_STDERR 287
LSF_INVOKE_CMD 287
LSF_JOB_STARTER 287
LSF_LIM_DEBUG 288
LSF_LOGDIR 289

LSF_MASTER 289
LSF_NIOS_DEBUG 289
LSF_NIOS_DIE_CMD 289
LSF_NIOS_IGNORE_SIGWINDOW 289
LSF_NIOS_PEND_TIMEOUT 289
LSF_RESOURCES 290
LSF_USE_HOSTEQUIV 290
LSF_USER_DOMAIN 290
environmental variables
LSB_EXEC_RUSAGE 275
LSB_NTRIES 282
EQUIV, lsf.cluster file 497
ERR_FILE, bacct -l 18
ESTIMATED FINISH TIME, bsla 144
/etc/hosts file 644
EVENT_ADRSV_FINISH record, lsb.acct 321
EVENT_UPDATE_INTERVAL, lsb.params file 379
EXCEPTION LOAD AND THRESHOLD, bhosts -l 54
EXCEPTION STATUS, bjobs -l 18, 67
EXCLUSIVE
bqueues -l 117
lsb.queues file 404
exclusive resource 492
EXEC_HOST, bjobs 64
EXEC_ON, bacct -b 17
Execution host, lsacct -l 188
EXINTERVAL, lsf.cluster file 481
EXIT
bjobs -A 68
bjobs -l 66
Exit status, lsacct -l 188
EXIT_RATE, lsb.hosts file 355
EXT_FILTER_PORT, lsf.licensescheduler file Parameters section, description 579
external_index, lsload 226

F

FAIRSHARE
bqueues -l 117
bqueues -r 122
lsb.queues file 404
FAIRSHARE_QUEUES
bqueues -l 118
lsb.queues file 405
FEATURE
blinfo output 84
blstat output 89
blusers output 95
Feature section, lsf.licensescheduler file, description 585
FILELIMIT
bqueues -l 115
lsb.queues file 406
files
adding default system lists 619
removing default system lists 619
viewing task lists 619
FINISH
bjgroup 59
bsla 144
FLEX_NAME, lsf.licensescheduler file Feature section, description 585
FLOAT_CLIENTS, lsf.cluster file 481
FLOAT_CLIENTS_ADDR_RANGE, lsf.cluster file 482
FLX_LICENSE_FILE, lsf.licensescheduler file Parameters section 580
FREE, blstat output 90
FREE CPU LIST, brlainfo 133
FREECPUS, brlainfo 132
FROM, bacct -b 17
FROM_HOST, bjobs 64

G

GOAL, bsla 143
GOALS, lsb.serviceclasses file 459
GROUP
blinfo output 85
lsf.licensescheduler file Feature section, description 589
lsf.licensescheduler file ProjectGroup section 593
GROUP_DISTRIBUTION, lsf.licensescheduler file Feature section 589
GROUP_MEMBER
lsb.hosts file 358
lsb.users file 466
GROUP_NAME
bjgroup 58
lsb.hosts file 358
lsb.users file 466

H

hierarchical fairshare user groups 467
HIST_HOURS, lsb.params file 380
HJOB_LIMIT, lsb.queues file 406
HOG_FACTOR, bacct -l 18
HOST, blusers output 96
host models, automatic detection 602
host types, automatic detection 602
HOST_CTRL record, lsb.events 333
HOST_INACTIVITY_LIMIT, lsf.cluster file 483
HOST_NAME
bhosts 51
lsb.hosts file 354
lshosts 217
lsload 224
lsmon 238
HOST_PARTITION_NAME, bhpart 56
HostExport section, lsb.resources 447
HOSTNAME
brlainfo 132
lsf.cluster file 491
HOSTRESORDER variable 644
HOSTS
bhpart 56
blimits 81
blinfo output 84
bqueues -l 119
lsb.hosts file 362
lsb.queues file 406
lsb.resources file Limit section 436
lsb.resources file ResourceReservation section 451

lsclusters 199
lsf.licensescheduler file Parameters section, description 580

hosts
exclusive resource 492
lost_and_found 51, 64
lsfinstall command 203

hosts file 303
hostsetup command, example 205
hostsetup script, lsfinstall command 205
HPART_NAME, lsb.hosts file 362

I

idle job exception
bacct -l -x 18
bjobs -l 67
bqueues -l 117

IDLE_FACTOR, bjobs -l 66
IGNORE_DEADLINE
bqueues -l 117
lsb.queues file 408

IMPT_JOBBKLG, lsb.queues file 408
Inact_Adm status, bqueues -l 113
Inact_Win status, bqueues -l 113
Inactive status
blsa 144
bqueues 111
bqueues -l 113

INCREASING, lsf.shared file 605
INPUT_FILE, bacct -l 18
install.config file
description 308
required variables 202

INSTALL_OPTION, win_install.config file 635
installation directory, lsfinstall command 203
INTERACTIVE, lsb.queues file 408
INTERRRUPTIBLE_BACKFILL, lsb.queues file 408
INTERVAL, lsf.shared file 605
Interval for a host to accept two jobs, bqueues -l 114
INUSE, blstat output 90
Involuntary con sw, lsacct 188
io
bqueues -l 116
lsb.hosts file 356
lsb.queues file 413
lsload 225

IRIX ULDB (User Limits Database)
description 572
jlimit.in file 572

it
bqueues -l 116
lsb.hosts file 356
lsb.queues file 413
lsload 225
lsmon 239

J

JL/H, bqueues 112
JL/P
bqueues 112
busers 181
lsb.users file 468

JL/U
bhosts 51
bqueues 112
lsb.hosts file 355

jlimit.in file, IRIX ULDB 572
JOB CONTROLS, bqueues -l 121
JOB EXCEPTION PARAMETERS, bqueues -l 116
job ID
limit 385
rollover 385
sequencing 385

JOB STATUS, bjobs -l 65
JOB_ACCEPT record, lsb.events 329
JOB_ACCEPT_INTERVAL
lsb.params file 380
lsb.queues file 409

JOB_ACTION_WARNING_TIME, lsb.queues file 410
JOB_ATTA_DATA record, lsb.events 344
JOB_ATTA_DIR, lsb.params file 380
JOB_CHUNK record, lsb.events 345
JOB_CLEAN record, lsb.events 342
JOB_CONTROLS, lsb.queues file 410
JOB_DEP_LAST_SUB, lsb.params file 381
JOB_EXECUTE record, lsb.events 341
JOB_EXIT_RATE_DURATION, lsb.params file 381
JOB_EXT_MSG record, lsb.events 344
JOB_FINISH record, lsb.acct 316
JOB_FLOW, bclusters 39
JOB_FORCE record, lsb.events 350
JOB_FORWARD record, lsb.events 328
JOB_IDLE, lsb.queues file 411
JOB_MODIFY2 record, lsb.events 337
JOB_MOVE record, lsb.events 332
JOB_NAME
bacct -b 17
bjobs 64

JOB_NEW record, lsb.events 325
JOB_OVERRUN, lsb.queues file 412
JOB_POSITION_CONTROL_BY_ADMIN, lsb.params file 381
JOB_PRIORITY_OVER_TIME, lsb.params file 382
JOB_REQUEUE record, lsb.events 342
JOB_SCHEDULING_INTERVAL, lsb.params file 382
JOB_SIGACT record, lsb.events 336
JOB_SIGNAL record, lsb.events 341
JOB_SPOOL_DIR, lsb.params file 382
JOB_START record, lsb.events 330
JOB_START_ACCEPT record, lsb.events 331
JOB_STARTER
bqueues -l 120
lsb.queues file 412

JOB_STATUS record, lsb.events 331
JOB_SWITCH record, lsb.events 332
JOB_TERMINATE_INTERVAL, lsb.params file 383
JOB_UNDERRUN, lsb.queues file 412
JOB_WARNING_ACTION, lsb.queues file 413

JOBID
- bacct -l 17
- bjobs 64
- bjobs -A 67
- blusers output 96

L

LIB_RECVTIMEOUT, lsf.licensescheduler file Parameters section 580
LIC_COLLECT, lsf.licensescheduler file 76
LIC_COLLECTOR, lsf.licensescheduler file ServiceDomain section, description 583
LIC_FLEX_API_ENABLE, lsf.licensescheduler file ServiceDomain section, description 584
LIC_SERVERS
- blinfo output 84
- lsf.licensescheduler file ServiceDomain section, description 583

LICENSE, lsb.resources file Limit section 437
LICENSE CLASS NEEDED, lshosts -l 218
license key, lsfinstall command 203
LICENSE_FILE, blinfo output 84
LICENSES_ENABLED, lshosts -l 218
lim.acct file 313
LIM_PORT, win_install.config file 635
LIMIT, lsf.licensescheduler file ProjectGroup section 594
Limit section, lsb.resources 434
LIMITS, blinfo output 85
limits, job ID 385
LM_LICENSE_FILE variable 271, 580
LM_REMOVE_INTERVAL, lsf.licensescheduler file Parameters section 580
LM_STAT_INTERVAL, lsf.licensescheduler file Parameters section 580
LMSTAT_PATH, lsf.licensescheduler file Parameters section 580
LOAD THRESHOLD, bhosts -l 54
load_index
- lsb.hosts file 356
- lsb.queues file 413

LOAD_INDEX record, lsb.events 335
LOAD_THRESHOLDS, lshosts -l 218
loadSched
- bhosts -l 53
- bjobs -l 65

loadStop
- bhosts -l 53
- bjobs -l 65

LOCAL, lsb.users file 470
local tasks in task files 620
LOCAL_DIR, win_install.config file 636
LOCAL_QUEUE, bclusters 39
LOCATION
- bhosts -s 55
- lsf.cluster file 495
- lshosts -s 219
- lsload -s 226

log files, nios.log.host_name 561
lost_and_found host 51, 64
lost_and_found queue 81
- bqueues 64

lost_and_found queue name, bqueues 111
ls
- bqueues -l 116
- lsb.hosts file 356
- lsb.queues file 413
- lsload 225
- lsmon 239

LS_ADMIN, setup.config file 625
LS_EXEC_T variable 272
LS_HOSTS, setup.config file 625
LS_JOBPID variable 272
LS_LICENSE_FILE, setup.config file 625
LS_LICENSE_SERVER_feature variable 272
LS_LMSTAT_PATH, setup.config file 625
LS_SUBCWD variable 272
LS_TOP, setup.config file 626
lsacct 186
lsacctmrg 189
lsadmin 190
lsb.acct file 315
lsb.events file 323
lsb.hosts file
- description 353
- time-based configuration 365

lsb.modules file 367
lsb.params file
- description 373
- SUB_TRY_INTERVAL parameter 387
- time-based configuration 396

lsb.queues file
- description 399
- time-based configuration 430

lsb.resources file
- description 433
- time-based configuration 455

lsb.serviceclasses file 457
lsb.users file
- description 465
- time-based configuration 472

LSB_API_CONNTIMEOUT, lsf.conf file 509
LSB_API_RECVTIMEOUT, lsf.conf file 509
LSB_BLOCK_JOBINFO_TIMEOUT, lsf.conf file 509
LSB_CHKPNT_DIR variable 272
LSB_CHUNK_RUSAGE, lsf.conf file 510
LSB_CMD_LOG_MASK, lsf.conf file 510
LSB_CMD_LOGDIR, lsf.conf file 511
LSB_CONFDIR, lsf.conf file 511
LSB_CPUSET_BESTCPUS, lsf.conf file 509
LSB_CRDIR, lsf.conf file 511
LSB_DEBUG
- lsf.conf file 512
- variable 273

LSB_DEBUG_CMD
- lsf.conf file 512
- variable 273

LSB_DEBUG_MBD
- lsf.conf file 513

variable 273
LSB_DEBUG_NQS
lsf.conf file 514
variable 273
LSB_DEBUG_SBD
lsf.conf file 514
variable 273
LSB_DEBUG_SCH
lsf.conf file 515
variable 273
LSB_DEFAULTPROJECT variable 273
LSB_DEFAULTQUEUE variable 274
LSB_DISABLE_RERUN_POST_EXEC, lsf.conf file 517
LSB_ECHKPNT_KEEP_OUTPUT, lsf.conf file 517
LSB_ECHKPNT_METHOD, lsf.conf file 517
LSB_ECHKPNT_METHOD_DIR, lsf.conf file 518
LSB_ERESTART_USRCMD variable 274
LSB_ESUB_METHOD, lsf.conf file 517, 518
LSB_EXEC_RUSAGE variable 275
LSB_EXECHOSTS variable 275
LSB_EXIT_PRE_ABORT variable 275
LSB_EXIT_REQUEUE variable 275
LSB_FRAMES variable 276
LSB_HCLOSE_BY_RES, LSF HPC extensions parameter 545
LSB_HOSTS variable 276
LSB_INTERACT_MSG_ENH, lsf.conf file 519
LSB_INTERACT_MSG_INTVAL, lsf.conf file 519
LSB_INTERACTIVE variable 276
LSB_IRIX_NODESIZE, lsf.conf file (OBSOLETE) 520
LSB_JOB_CPULIMIT, lsf.conf file 520
LSB_JOB_MEMLIMIT, lsf.conf file 521
LSB_JOB_STARTER variable 276
LSB_JOBEXIT_INFO variable 277
LSB_JOBEXIT_STAT variable 278
LSB_JOBFILENAME variable 278
LSB_JOBID variable 278
LSB_JOBINDEX variable 279
LSB_JOBINDEX_STEP variable 279
LSB_JOBNAME variable 279
LSB_JOBPEND variable 280
LSB_JOBPGIDS variable 280
LSB_JOBPIDS variable 280
LSB_KEEP_SYSDEF_RLIMIT, lsf.conf file 520
LSB_LOCALDIR, lsf.conf file 523
LSB_MAILPROG, lsf.conf file 523
LSB_MAILSERVER, lsf.conf file 524
LSB_MAILSIZE variable 280
LSB_MAILSIZE_LIMIT, lsf.conf file 524
LSB_MAILTO, lsf.conf file 525
LSB_MAX_JOB_DISPATCH_PER_SESSION, lsf.conf file 525
LSB_MAX_NQS_QUEUES, lsf.conf file 526
LSB_MAX_PROBE_SBD, lsf.conf file 526
LSB_MBD_PORT, lsf.conf file 526, 555
LSB_MC_CHKPNT_RERUN, lsf.conf file 526
LSB_MC_INITFAIL_MAIL, lsf.conf file 527
LSB_MC_INITFAIL_RETRY, lsf.conf file 527
LSB_MCPU_HOSTS variable 281
LSB_MEMLIMIT_ENFORCE, lsf.conf file 527
LSB_MIG2PEND, lsf.conf file 527
LSB_MOD_ALL_JOBS, lsf.conf file 528
LSB_NCPU_ENFORCE, lsf.conf file 528
LSB_NQS_PORT, lsf.conf file 528
LSB_NQS_PORT variable 282
LSB_NTRIES environment variable 387
LSB_NTRIES variable 282
LSB_OLD_JOBID variable 282
LSB_OUTPUT_TARGETFAILED variable 282
LSB_PRE_POST_EXEC_USER, lsf.sudoers file 613
LSB_PSET_BIND_DEFAULT, lsf.conf file 529
LSB_QUERY_PORT, lsf.conf file 529
LSB_QUEUE variable 282
LSB_REMOTEINDEX variable 283
LSB_REMOTEJID variable 283
LSB_REQUEUE_TO_BOTTOM, lsf.conf file 530
LSB_RESTART variable 283
LSB_RESTART_PGID variable 283
LSB_RESTART_PID variable 283
LSB_RLA_HOST_LIST, lsf.conf file 530
LSB_RLA_PORT, lsf.conf file 530
LSB_RLA_UPDATE, lsf.conf file 530
LSB_RLA_WORKDIR, lsf.conf file 531
LSB_RMS_MAXNUMNODES 531
LSB_RMS_MAXNUMRAILS, lsf.conf file 531
LSB_RMS_MAXPTILE, lsf.conf file 531
LSB_RMSACCT_DELAY, lsf.conf file 531
LSB_SBD_PORT
lsf.conf file 532, 555
lsf.conf file (INTERNAL) 532
LSB_SET_TMPDIR, lsf.conf file 532
LSB_SHAREDIR, lsf.conf file 532
LSB_SHORT_HOSTLIST, lsf.conf file 532
LSB_SIGSTOP, lsf.conf file 533
LSB_SLURM_BESTFIT, lsf.conf file 532
LSB_STDOUT_DIRECT, lsf.conf file 534
LSB_SUB_CLUSTER variable 284
LSB_SUB_COMMAND_LINE variable 284
LSB_SUB_COMMANDNAME, lsf.conf file 533
LSB_SUB_EXTSCHED_PARAM variable 284
LSB_SUB_JOB_ACTION_WARNING_TIME variable 284
LSB_SUB_JOB_WARNING_ACTION variable 284
LSB_SUB_PARM_FILE variable 284
LSB_SUSP_REASONS variable 284
LSB_SUSP_SUBREASONS variable 285
LSB_TIME_CMD, lsf.conf file 534
LSB_TIME_MBD, lsf.conf file 534
LSB_TIME_RESERVE_NUMJOBS, lsf.conf file 535
LSB_TIME_SBD, lsf.conf file 535
LSB_TIME_SCH, lsf.conf file 535
LSB_UTMP, lsf.conf file 535
lsclusters 198
lseligible 200
LSF 569
LSF administrator, lsfinstall command 203
lsf.cluster file 480
lsf.cluster_name.license.acct file 499

lsf.conf file 504
lsf.licensescheduler file, reference 577
lsf.shared file 599
lsf.sudoers file 608
lsf.task file 617
lsf.task.cluster file 617
LSF_ADD_CLIENTS, install.config file 309
LSF_ADD_SERVERS, install.config file 309
LSF_ADMINS
  install.config file 309
  slave.config file 629
LSF_AFS_CELLNAME, lsf.conf file 536
LSF_AM_OPTIONS, lsf.conf file 536
LSF_API_CONNTIMEOUT, lsf.conf file 536
LSF_API_RECVTIMEOUT, lsf.conf file 536
LSF_AUTH, lsf.conf file 537
LSF_AUTH_DAEMONS, lsf.conf file 537
LSF_BINDIR
  cshrc.lsf and profile.lsf files 301
  lsf.conf file 537
LSF_CLIENTS, win_install.config file 636
LSF_CLUSTER_NAME
  install.config file 310
  win_install.config file 637
LSF_CMD_LOG_MASK, lsf.conf file 538
LSF_CMD_LOGDIR
  lsf.conf file 538
  variable 285
LSF_CONF_RETRY_INT, lsf.conf file 539
LSF_CONF_RETRY_MAX, lsf.conf file 539
LSF_CONFDIR, lsf.conf file 539
LSF_DAEMON_WRAP, lsf.conf file 539
LSF_DEBUG_CMD variable 286
LSF_DEBUG_LIM
  lsf.conf file 539
  variable 286
LSF_DEBUG_RES
  lsf.conf file 540
  variable 286
LSF_DESERVE, blstat output 89
LSF_DHCP_ENV, lsf.conf file 541
LSF_DISPATCHER_LOGDIR, lsf.conf file 541
LSF_DUALCORE product name, lsf.cluster_name.license.acct file 501
lsf_dualcore_x86 license feature 543
LSF_DYNAMIC_HOST_WAIT_TIME, lsf.conf file 310, 542
LSF_DYNAMIC_SERVERS, win_install.config file 636
LSF_EAUTH_AUX_DATA variable 286
LSF_EAUTH_AUX_PASS variable 286
LSF_EAUTH_CLIENT variable 286
LSF_EAUTH_KEY, lsf.sudoers file 613
LSF_EAUTH_SERVER variable 287
LSF_EAUTH_UID variable 287
LSF_EAUTH_USER, lsf.sudoers file 613
LSF_EEXEC_USER, lsf.sudoers file 614
LSF_ELIM_DEBUG, lsf.cluster file 484
LSF_ELIM_RESTARTS, lsf.cluster file 485
LSF_ENABLE_CSA, lsf.conf file 542
LSF_ENABLE_DUALCORE, lsf.conf file 543
LSF_ENABLE_EXTSCHEDULER
  bsub 157
  lsf.conf file 544
LSF_ENVDIR, lsf.conf file 301, 544
LSF_EVENT_PROGRAM, lsf.conf file 544
LSF_EVENT_RECEIVER, lsf.conf file 544
LSF_FREE, blstat output 89
LSF_HOST_ADDR_RANGE, lsf.cluster file 485
LSF_HPC_EXTENSIONS, lsf.conf file 544
LSF_HPC_NCPU_COND, lsf.conf file 548
LSF_HPC_NCPU_INCR_CYCLES, lsf.conf file 548
LSF_HPC_NCPU_INCREMENT, lsf.conf file 548
LSF_HPC_NCPU_THRESHOLD, lsf.conf file 548
LSF_HPC_PJL_LOADENV_TIMEOUT, lsf.conf file 549
LSF_ID_PORT, lsf.conf file 549
LSF_INCLUDEDIR, lsf.conf file 549
LSF_INDEP, lsf.conf file 549
LSF_INTERACTIVE_STDERR
  lsf.conf file 550
  variable 287
LSF_INVOKE_CMD variable 287
LSF_IRIX_BESTCPUS, lsf.conf file (OBSOLETE) 551
LSF_JOB_STARTER variable 287
LSF_LD_SECURITY, lsf.conf 551
LSF_LIBDIR
  cshrc.lsf and profile.lsf files 301
  lsf.conf file 551
LSF_LIC_SCHED_HOSTS, lsf.conf file 551
LSF_LIC_SCHED_PREEMPT_REQUEUE, lsf.conf file 551
LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, lsf.conf file 552
LSF_LIC_SCHED_PREEMPT_STOP, lsf.conf file 552
LSF_LICENSE, install.config file 311
LSF_LICENSE_ACCT_PATH, lsf.conf file 552
LSF_LICENSE_FILE, lsf.conf file 553
LSF_LICENSE_NOTIFICATION_INTERVAL, lsf.conf file 553
LSF_LIM_DEBUG
  lsf.conf file 554
  variable 288
LSF_LIM_IGNORE_CHECKSUM, lsf.conf file 554
LSF_LIM_PLUGINDIR, lsf.conf file 555
LSF_LIM_PORT
  lsf.conf file 555
  slave.config file 629
LSF_LIM_SOL27_PLUGINDIR, lsf.conf file 555
LSF_LOAD_PLUGINS, lsf.sudoers file 614
LSF_LOCAL_RESOURCES
  lsf.conf file 556
  slave.config file 630
LSF_LOG_MASK, lsf.conf file 556, 557
LSF_LOGDIR
  lsf.conf file 558, 559
  variable 289
LSF_MACHDEP, lsf.conf file 559
LSF_MANAGER product name, lsf.cluster_name.license.acct file 500
LSF_MANDIR, lsf.conf file 560
LSF_MASTER variable 289

LSF_MASTER_LIST
install.config file 311
lsf.conf file 560
LSF_MC_NON_PRIVILEGED_PORTS, lsf.conf file 560
LSF_MISC, lsf.conf file 561
LSF_MULTICLUSTER product name, lsf.cluster_name.license.acct file 500
LSF_NIOS_DEBUG
lsf.conf file 561
variable 289
LSF_NIOS_DIE_CMD variable 289
LSF_NIOS_IGNORE_SIGWINDOW variable 289
LSF_NIOS_JOBSTATUS_INTERVAL, lsf.conf file 562
LSF_NIOS_PEND_TIMEOUT variable 289
LSF_NIOS_RES_HEARTBEAT, lsf.conf file 562
LSF_NON_PRIVILEGED_PORTS, lsf.conf file 561
LSF_PAM_HOSTLIST_USE, lsf.conf file 563
LSF_PAM_PLUGINDIR, lsf.conf file 563
LSF_PAM_USE_ASH, lsf.conf file 563
LSF_PIM_INFODIR, lsf.conf file 564
LSF_PIM_SLEEPTIME, lsf.conf file 564
LSF_PIM_SLEEPTIME_UPDATE, lsf.conf file 564
LSF_POE_TIMEOUT_BIND, lsf.conf file 563
LSF_POE_TIMEOUT_SELECT, lsf.conf file 564
LSF_QUIET_INST, install.config file 311
LSF_RES_ACCT, lsf.conf file 565
LSF_RES_ACCTDIR, lsf.conf file 565
LSF_RES_CONNECT_RETRY, lsf.conf file 565
LSF_RES_DEBUG, lsf.conf file 566
LSF_RES_PLUGINDIR, lsf.conf file 566
LSF_RES_PORT, lsf.conf file 555
LSF_RES_RLIMIT_UNLIM, lsf.conf file 566
LSF_RES_SOL27_PLUGINDIR, lsf.conf file 567
LSF_RES_TIMEOUT, lsf.conf file 567
LSF_RESOURCES variable 290
LSF_ROOT_REX, lsf.conf file 567
LSF_RSH, lsf.conf file 567
LSF_SECUREDIR, lsf.conf file 568
LSF_SERVER_HOSTS
lsf.conf file 568
slave.config file 629
LSF_SERVERDIR
cshrc.lsf and profile.lsf files 302
lsf.conf file 568
LSF_SERVERS, win_install.config file 637
LSF_SHELL_AT_USERS, lsf.conf file 568
LSF_SHIFT_JIS_INPUT, lsf.conf file 569
LSF_SLURM_DISABLE_CLEANUP, lsf.conf file 569
LSF_SLURM_TMPDIR, lsf.conf file 569
LSF_STARTUP_PATH, lsf.sudoers file 615
LSF_STARTUP_USERS, lsf.sudoers file 614
LSF_STRICT_CHECKING, lsf.conf file 569
LSF_STRIP_DOMAIN, lsf.conf file 570
LSF_TARDIR
install.config file 312
slave.config file 630
LSF_TIME_CMD, lsf.conf file 570
LSF_TIME_LIM, lsf.conf file 571
LSF_TIME_RES, lsf.conf file 571
LSF_TMPDIR, lsf.conf file 571
LSF_TOP
install.config file 312
slave.config file 631
win_install.config file 637
LSF_TOPD_PORT, lsf.conf file 572
LSF_TOPD_WORKDIR, lsf.conf file 572
LSF_ULDB_DOMAIN, lsf.conf file 572
LSF_USE, blstat output 89
LSF_USE_HOSTEQUIV
lsf.conf file 574
variable 290
LSF_USER_DOMAIN
lsf.conf file 574
variable 290
LSF_VPLUGIN, lsf.conf file 575
lsf6.2_lsfinstall.tar.Z file 203
lsfinstall command
description 202
location 203
lsfmon 208
lsfrestart 209
lsfshutdown 211
lsfstartup 212
.lsftask file 617
lsgrun 213
lshosts 216
lsid 220
lsinfo 221
lsload 223
lsloadadj 228
lslockhost, obsolete command. *See* lsadmin limlock
lslogin 230
lsltasks 232
lsmake 234
lsmon 236
lspasswd 240
lsplace 241
lsrcp 243, 244
lsreconfig, obsolete command. *See* lsadmin reconfig
lsrtasks 246
lsrun 248
lstcsh 251
lsunlockhost, obsolete command. *See* lsadmin limunlock

M

mail, configuring on UNIX 523
MANDATORY_EXTSCHED
bsub 157
lsb.queues file 414
master host candidates, lsfinstall command 203
MASTER_HOST, lsclusters 198
MASTER_INACTIVITY_LIMIT, lsf.cluster file 487
MAX
bhosts 52
bqueues 112
busers 181
MAX_ACCT_ARCHIVE_FILE, lsb.params file 384

MAX_CONCURRENT_JOB_QUERY, lsb.params file 384
MAX_GROUPS, lsbatch.h file 466
MAX_INFO_DIRS, lsb.params file 384
MAX_JOB_ARRAY_SIZE, lsb.params file 385
MAX_JOB_ATTA_SIZE, lsb.params file 385
MAX_JOB_MSG_NUM, lsb.params file 386
MAX_JOB_NUM, lsb.params file 386
MAX_JOBID, lsb.params file 385
MAX_JOBINFO_QUERY_PERIOD, lsb.params file 386
MAX_JOBS, lsb.users file 468
MAX_PEND_JOBS
    lsb.params file 387
    lsb.users file 469
MAX_PREEXEC_RETRY, lsb.params file 387
MAX_RSCHED_TIME, lsb.queues file 414
MAX_SBD_CONNS, lsb.params file 387
MAX_SBD_FAIL, lsb.params file 387
MAX_SCHED_STAY, lsb.params file (OBSOLETE) 388
MAX_USER_PRIORITY, lsb.params file 388
maximum, job ID 385
Maximum slot reservation time, bqueues -l 120
maxmem, lshosts 217
maxswp, lshosts 218
maxtmp, lshosts -l 218
MBD_DIE record, lsb.events 334
MBD_REFRESH_TIME, lsb.params file 388
MBD_SLEEP_TIME, lsb.params file 389
MBD_START record, lsb.events 334
MC_PENDING_REASON_PKG_SIZE, lsb.params file 389
MC_PENDING_REASON_UPDATE_INTERVAL, lsb.params file 390
MC_RECLAIM_DELAY, lsb.params file 389
MC_RUSAGE_UPDATE_INTERVAL, lsb.params file 390
MEM
    bacct -l 18
    bjobs -l 66
    blimits 82
    lsb.resources file HostExport section 448
    lsb.resources file Limit section 437
mem
    bqueues -l 116
    lsb.hosts file 356
    lsb.queues file 413
    lsload 225
    lsmon 239
MEMLIMIT
    bqueues -l 114
    lsb.queues file 414
    per parallel task 546
    per-job limit 521
Messages rcvd, lsacct 188
Messages sent, lsacct 188
METHOD, lsb.resources file ReservationUsage section 454
MIG
    lsb.hosts file 355
    lsb.queues file 415
MIG record, lsb.events 337
Migration threshold, bqueues -l 113
MIN_SWITCH_PERIOD, lsb.params file 390
model
    lsf.cluster file 492
    lshosts 217
MODELNAME, lsf.shared file 602
MPEND, busers 182
MultiCluster account mapping 470
MXJ, lsb.hosts file 356

N

NAME
    blimits 81
    lsb.resources file Limit section 438
    lsb.resources file ResourceReservation section 452
    lsb.resources file SharedResourceExport section 450
    lsb.serviceclasses file 460
    lsf.licensescheduler file Feature section, description 585
    lsf.licensescheduler file ServiceDomain section, description 583
NCPU/NODE NSTATIC_CPUSETS, brlainfo 132
NCPUS
    bacct -U 19
    brlainfo 132
ncpus, lshosts 217
nd, lsf.cluster file 492
ndisks, lshosts -l 218
NEW_JOB_SCHED_DELAY, lsb.queues file 416
NFREECPUS ON EACH NODE, brlainfo 133
NFS (Network File System) automount 642
NHOSTS, lsb.resources file HostExport section 447
NICE
    bqueues -l 113
    lsb.queues file 416
NINSTANCES, lsb.resources file SharedResourceExport section 450
NIOS, standard message format 551
nios.log.host_name 561
NJOBS
    bhosts 52
    bjgroup 58
    bjobs -A 68
    bqueues 112
    bsla 144
    busers 182
NLICS, blusers output 96
NNODES, brlainfo 132
NO_INTERACTIVE, bqueues -l 117
NON_LSF_DESERVE, blstat output 90
NON_LSF_FREE, blstat output 90
NON_LSF_USE, blstat output 89
NON_SHARED
    blinfo output 85
    blstat output 90
    lsf.licensescheduler file ProjectGroup section 594
NON_SHARED_DISTRIBUTION, lsf.licensescheduler file Feature section, description 590
NON-SHARED_DISTRIBUTION, blinfo output 85
NQS DESTINATION QUEUES, bqueues -l 119

NQS_QUEUES, lsb.queues file 416
NQS_QUEUES_FLAGS, lsb.params file 391
NQS_REQUESTS_FLAGS, lsb.params file 391
NSTATIC_CPUSETS, brlainfo 132
NTASKS, blusers output 96
NTHREAD, bjobs -l 66

O

obsolete commands
    bqc. *See* badmin qclose
    breboot. *See* badmin reconfig
    breconfig. *See* badmin reconfig
    lslockhost. *See* lsadmin limlock
    lsreconfig. *See* lsadmin reconfig
    lsunlockhost. *See* lsadmin limunlock
OK license usage status
    bld.license.acct file 296
    lsf.cluster_name.license.acct file 501
ONLY_INTERACTIVE, bqueues -l 117
Open status, bqueues 111
OPTIMUM NUMBER OF RUNNING JOBS, bsla 144
OTHERS
    blstat output 89
    blusers output 96
OUTPUT_FILE, bacct -l 18
overrun job exception
    bacct -l -x 18
    bjobs -l 67
    bqueues -l 117
OVERUSE license usage status
    bld.license.acct file 296
    lsf.cluster_name.license.acct file 501
OWNER, bjobs -A 68
OWNERSHIP
    blinfo output 85
    lsf.licensescheduler file ProjectGroup section, description 594

P

Page faults, lsacct 187
PARALLEL_SCHED_BY_SLOT 391
Parameters section, lsf.licensescheduler file, description 578
PARAMETERS/STATISTICS, bqueues -l 113
PEND
    bhist 46
    bjgroup 58
    bjobs -A 68
    bjobs -l 65
    bqueues 112
    bsla 144
    busers 182
PEND_REASON_UPDATE_INTERVAL 392
PENDING REASONS, bjobs -l 65
PER_HOST
    lsb.resources file HostExport section 447
    lsb.resources file Limit section 438
PER_PROJECT, lsb.resources file Limit section 439
PER_QUEUE, lsb.resources file Limit section 439
PER_USER, lsb.resources file Limit section 440
pg
    bqueues -l 116
    lsb.hosts file 356
    lsb.queues file 413
    lsload 225
    lsmon 238
PG_SUSP_IT, lsb.params file 392
PGID, bjobs -l 66
PID, lsacct -l 188
PIDS, blusers output 96
PIDs, bjobs -l 67
PJOB_LIMIT, lsb.queues file 417
PORT
    blinfo output 84
    lsf.licensescheduler file Parameters section 581
POST_EXEC
    bqueues -l 120
    lsb.queues file 417
PRE_EXEC
    bqueues -l 119
    lsb.queues file 418
PRE_EXEC_START record, lsb.events 349
PREEMPT_FINISH_TIME, lsb.params file 391
PREEMPT_FOR, lsb.params file 392
PREEMPT_LSF, lsf.licensescheduler file Feature section, description 590
PREEMPT_RESERVE, lsf.licensescheduler file Feature section, description 591
PREEMPT_RUN_TIME, lsb.params file 390
PREEMPTABLE, bqueues -l 121
PREEMPTABLE_RESOURCES, lsb.params file 392
PREEMPTION
    bqueues -l 120
    lsb.queues file 418
PREEMPTION_WAIT_TIME, lsb.params file 393
PREEMPTIVE, bqueues -l 120
primary LSF administrator, lsfinstall command 203
PRIO, bqueues 111
PRIORITY
    bhpart 56
    bsla 143
    lsb.queues file 419
    lsb.serviceclasses file 460
    lsf.licensescheduler file Projects section, description 596
PROBE_TIMEOUT, lsf.cluster file 487
PROCESSLIMIT
    bqueues -l 114
    lsb.queues file 419
PROCLIMIT
    bqueues -l 114
    lsb.queues file 420
PRODUCTS, lsf.cluster file 487
profile.lsf file
    description 298
    setting the LSF environment 299
PROJECT
    blstat output 90
    blusers output 96
Project

bhist -l 46
bjobs -l 65
PROJECT/GROUP, blstat output 91
PROJECT_NAME, bacct -l 17
ProjectGroup section, lsf.licensescheduler file, description 593
PROJECTS
blimits 81
lsb.resources file Limit section 440
lsf.licensescheduler file Projects section 596
Projects section, lsf.licensescheduler file, description 596
PSUSP
bhist 46
bjobs -A 68
bjobs -l 65

Q

QJOB_LIMIT, lsb.queues file 420
QUEUE
bacct -b 17
bjobs 64
QUEUE_CTRL record, lsb.events 333
QUEUE_NAME
bqueues 111
lsb.queues file 420
QUEUES
blimits 81
lsb.resources file Limit section 441
queues, lost_and_found 64, 81

R

r15m
bqueues -l 116
lsb.hosts file 356
lsb.queues file 413
lsload 225
lsmon 238
r15s
bqueues -l 115
lsb.hosts file 356
lsb.queues file 413
lsload 225
lsmon 238
r1m
bqueues -l 116
lsb.hosts file 356
lsb.queues file 413
lsload 225
lsmon 238
RB_PLUGIN, lsb.modules file 370
RCVJOBS_FROM, lsb.queues file 421
RECEIVE_JOBS_FROM, bqueues -l 120
RECV_FROM, lsf.cluster file 498
RELEASE, lsf.shared file 605
REMOTE
bclusters 39
lsb.users file 470
remote shell, lsrcp 244
remote task list 618
remote tasks in task files 620
REMOTE_CLUSTER, bclusters 40
REQUEUE_EXIT_VALUES
bqueues -l 120
lsb.queues file 421
required install.config and slave.config variables 202
RERUNNABLE
bqueues -l 121
lsb.queues file 421
RES_REQ
bqueues -l 120
lsb.queues file 422
RES_SELECT, lsb.resources file HostExport section 447
ReservationUsage section, lsb.resources 454
RESERVE, blstat output 90
RESERVE_BY_STARTTIME, LSF HPC extensions parameter 545
RESERVED
bhosts -s 55
bhpart 57
RESOURCE
bhosts -s 55
blusers output 96
lsb.resources file Limit section 441
lsb.resources file ReservationUsage section 454
lshosts -s 219
lsload -s 226
RESOURCE LIMITS
bjobs -l 67
bqueues -l 114
RESOURCE USAGE, bjobs -l 66
Resource usage of tasks selected, lsacct 187
RESOURCE_FLOW, bclusters 40
RESOURCE_RESERVE, lsb.queues file 422
RESOURCE_RESERVE_PER_SLOT, lsb.params file 393
RESOURCENAME
lsf.cluster file 496
lsf.shared file 604
ResourceReservation section, lsb.resources 451
RESOURCES
lsf.cluster file 492
lshosts 218
RESUME_COND
bqueues -l 120
lsb.queues file 423
RETRY_LIMIT, lsf.cluster file 487
REXPRI, lsf.cluster file 492
rexpri, lshosts -l 218
.rhosts file 644
rhostsetup script, lsfinstall command 205
rollover, job IDs 385
rsh command
badmin hstartup all 29
lsadmin limstartup all 191
lsadmin resstartup all 192
lsfrestart lsfshutdown lsfstartup 209, 211, 212
lsrcp 244
RSV
bhosts 52
bqueues -l 113

busers 182
RSV_HOSTS, bacct -U 19
RSVID, bacct -U 18
RUN
bhist 46
bhosts 52
bjgroup 59
bjobs -A 68
bjobs -l 65
bqueues 113
bsla 144
busers 182
RUN_JOB_FACTOR, lsb.params file 394
RUN_TIME, bhpart 57
RUN_TIME_FACTOR, lsb.params file 394
RUN_WINDOW, lsb.queues file 423
RUN_WINDOWS
bqueues -l 118
lshosts -l 218
RUNLIMIT
bqueues -l 115
lsb.queues file 424
RUNWINDOW, lsf.cluster file 493
RUSAGE, blusers output 96

S

SBD_SLEEP_TIME, lsb.params file 394
SBD_UNREPORTED_STATUS record, lsb.events 345
SCH_DISABLE_PHASES, lsb.modules file 370
SCH_PLUGIN, lsb.modules file 368
SCHED_INTERVAL, lsf.licensescheduler file 581
Schedule delay for a new job, bqueues -l 113
SCHEDULING PARAMETERS, bqueues -l 115
SCHEDULING POLICIES
bqueues -l 117
bqueues -r 122
secure shell 244
security, daemons, increasing 569
SEND_JOBS_TO, bqueues -l 120
sendmail program 523
server
lsf.cluster file 493
lshosts 218
server hosts, lsfinstall command 203
SERVER_HOST, win_install.config file 638
SERVERS, lsclusters 199
Servers, lsf.shared file 600
service class, examples 461
SERVICE CLASS NAME, bsla 143
SERVICE_ACCT, win_install.config file 638
SERVICE_DOMAIN
blinfo output 84
blstat output 89
blusers output 95, 96
SERVICE_DOMAINS, lsf.licensescheduler file Feature section, description 591
ServiceDomain section, lsf.licensescheduler file, description 583
setuid permissions 644
setup.config file 624
seven-digit job ID 385
SHARE, blstat output 90
SHARE_INFO_FOR, blstat output 90
shared files 642
SharedResourceExport section, lsb.resources 450
SHARES
bhpart 56
blinfo output 85
lsf.licensescheduler file ProjectGroup section, description 594
SHORT_EVENTFILE, LSF HPC extensions parameter 545
SLA scheduling, service classes, examples 461
SLA THROUGHPUT, bsla 144
slave.config file 628
required variables 202
SLOT_POOL
bqueues -l 121
lsb.queues file 425
SLOT_RESERVE, lsb.queues file 425
SLOT_SHARE
bqueues -l 121
lsb.queues file 426
SLOTS
blimits 82
lsb.resources file HostExport section 448
lsb.resources file Limit section 442
SLOTS_PER_PROCESSOR, lsb.resources file Limit section 443
SNDJOBS_TO, lsb.queues file 426
ssh command
badmin hstartup all 29
lsadmin limstartup all 191
lsadmin resstartup all 192
lsfrestart lsfshutdown lsfstartup 209, 211, 212
lsrcp 244
SSUSP
bhist 46
bhosts 52
bjgroup 59
bjobs -A 68
bjobs -l 65
bqueues -l 113
bsla 144
busers 182
STACKLIMIT
bqueues -l 115
lsb.queues file 426
START_TIME, blusers output 96
STARTED, bhpart 57
Starting time, lsacct -l 188
STAT, bjobs 64
STATIC CPUSETS, brlainfo 133
STATUS
bacct -l 17
bclusters 39, 40
bhosts 51
bhosts -l 53
bqueues 111
bqueues -l 113

bsla 144
lsclusters 198
status
lsload 224
lsmon 238
STOP_COND
bqueues -l 120
lsb.queues file 426
SUB_TRY_INTERVAL, lsb.params file 394
SUB_TRY_INTERVAL parameter in lsb.params 387
SUBMIT_TIME
bacct -b 17
bjobs 65
SUSP, bqueues 113
SUSPENDING REASONS, bjobs -l 65
SWAP
bacct -l 18
bjobs -l 66
lsb.resources file HostExport section 448
SWAPLIMIT
bqueues -l 114
lsb.queues file 427
per parallel task 546
Swaps, lsacct 187
SWP
blimits 82
lsb.resources file Limit section 444
swp
bqueues -l 116
lsb.hosts file 356
lsb.queues file 413
lsload 225
lsmon 239
SYSTEM_MAPPING_ACCOUNT, lsb.params file 395

T

task files
description 617
format 620
permissions 619
sections 620
task lists
files 618
remote 618
viewing 619
TASK_MEMLIMIT, LSF HPC extensions parameter 546
TASK_SWAPLIMIT, LSF HPC extensions parameter 546
taskman 260
TERMINATE_WHEN, lsb.queues file 427
THREADLIMIT
bqueues -l 115
lsb.queues file 428
THROUGHPUT, bsla 144
Time range of ended tasks, lsacct 187
Time range of started tasks, lsacct 187
TIME_WINDOW
bacct -U 19
lsb.resources file ResourceReservation section 452
time-based configuration
lsb.hosts 365
lsb.params 396
lsb.queues 430
lsb.resources 455
lsb.users 472
TMP
blimits 82
lsb.resources file Limit section 445
tmp
bqueues -l 116
lsb.hosts file 356
lsb.queues file 413
lsload 225
lsmon 239
/tmp_mnt directory 642
top-level installation directory (LSF_TOP) 203
TOTAL
bhist 46
bhosts -s 55
blinfo output 84
Total number of tasks, lsacct 187
TOTAL_FREE, blstat output 89
TOTAL_INUSE, blstat output 89
TOTAL_RESERVE, blstat output 89
TURNAROUND, bacct -b 17
Turnaround, lsacct 188
TYPE
bacct -U 18
lsb.resources file HostExport section 448
lsf.shared file 604
type
lsf.cluster file 493
lshosts 217
TYPENAME, lsf.shared file 601

U

U/UID, bacct -b 17
UJOB_LIMIT, lsb.queues file 428
ULDB (IRIX User Limits Database)
description 572
jlimit.in file 572
underrun job exception
bacct -l -x 18
bjobs -l 67
bqueues -l 117
UNFULFILL record, lsb.events 335
UNKNOWN, blusers output 96
UNKWN
bhist 46
bjobs -l 66
untrusted environments 569
USER
bacct -U 19
bjobs 64
blusers output 95, 96
user account mapping in MultiCluster 470
user and host name, lsacct -l 188
USER GROUP, bsla 143
user groups
hierarchical fairshare 467
maximum number 466

User section, lsb.users file 468
USER/GROUP
    bhpart 56
    busers 181
USER_NAME, lsb.users file 468
USER_SHARES
    bqueues -l 118
    lsb.hosts file 363
    lsb.users file 467
UserGroup section, lsb.users file 466
UserMap section, lsb.users file 470
USERS
    blimits 81
    bqueues -l 119
    lsb.queues file 428
    lsb.resources file Limit section 445
    lsb.resources file ResourceReservation section 453
    lsb.serviceclasses file 461
USUSP
    bhist 46
    bhosts 52
    bjgroup 59
    bjobs -A 68
    bjobs -l 65
    bqueues -l 113
    bsla 144
    busers 182
ut
    bqueues -l 116
    lsb.hosts file 356
    lsb.queues file 413
    lsload 225
    lsmon 238

V

VALUE
    lshosts -s 219
    lsload -s 226
variables. *See* environment variables
Voluntary cont sw, lsacct 188

W

WAIT
    bacct -b 17
    bjobs -l 66
wgpasswd 261
wguser 263
win_install.config file 634
WORKLOAD_DISTRIBUTION, lsf.licensescheduler file
        Feature section, description 591

X

XLSF_APPDIR, lsf.conf file 575
XLSF_UIDDIR
    cshrc.lsf and profile.lsf files 302
    lsf.conf file 576

Z

ZOMBI, bjobs -l 66